Faster Adaptive Network Based Fuzzy Inference System

Submitted in partial fulfillment

Of the requirements for

The Degree of Doctor of Philosophy

In Statistics

At the

University of Canterbury

By

Issarest Weeraprajak

_____

University of Canterbury

2007

CONTENTS

Abstract

It has been shown by Roger Jang in his paper titled "Adaptive-network-based fuzzy inference systems" that the Adaptive Network based Fuzzy Inference System can model nonlinear functions, identify nonlinear components in a control system, and predict a chaotic time series. The system use hybrid-learning procedure which employs the back-propagation-type gradient descent algorithm and the least squares estimator to estimate parameters of the model.

However the learning procedure has several shortcomings due to the fact that

- There is a harmful and unforeseeable influence of the size of the partial derivative on the weight step in the back-propagation-type gradient descent algorithm.
- In some cases the matrices in the least square estimator can be ill-conditioned.
- Several estimators are known which dominate, or outperform, the least square estimator.

Therefore this thesis develops a new system that overcomes the above problems, which is called the "Faster Adaptive Network Fuzzy Inference System" (FANFIS). The new system in this thesis is shown to significantly out perform the existing method in predicting a chaotic time series , modelling a three-input nonlinear function and identifying dynamical systems.

We also use FANFIS to predict five major stock closing prices in New Zealand namely Air New Zealand "A" Ltd., Brierley Investments Ltd., Carter Holt Harvey Ltd., Lion Nathan Ltd. and Telecom Corporation of New Zealand Ltd. The result shows that the new system out performed other competing models and by using simple trading strategy, profitable forecasting is possible.

Acknowledgments

I would like to thank

My Lord and my Saviour Jesus Christ for being my all,

My late dad Admiral Charn Weeraprajak for his unconditional love,

Dr. Easaw Chacko my supervisor for his wisdom and guidance,

Dr. Peter Griffith for his wise advice,

Julie and John Morrow for their unending hospitality,

Paul and Nicky Clarks for being the greatest guardians,

My mum, my sister and my friends for their support,

With out them I can not possibly finish this thesis.

"I will lift up my eyes unto the hills, from whence cometh my help.

My help cometh from the Lord, which made heaven and earth."

The book of Psalm chapter 121 verses 1:2

In King James Version of the Bible

Papers Resulting from this Thesis

Conference presentations

- New learning algorithm for adaptive based fuzzy inference systems in application of forecasting chaotic time series (2006), Presented at International Conference on Time Series Econometrics, Finance and Risk, 29 June - 1 July, 2006 in Perth, Australia.
- Faster Adaptive Network Based Fuzzy Inference System (2007), Paper presented at International Conference on Modelling and Optimization of Structures, Processes and Systems, 22-24 January, 2007 in Durban, South Africa.

Chapter 1

1 Introduction

Adaptive Network Based Fuzzy Inference System (ANFIS) [1] is an integrated system using the fuzzy inference system [2, 13] and the adaptive networks hybrid learning procedure. This learning procedure employs the back-propagation-type gradient descent algorithm [3, 4] and the least squares estimator (LSE) to estimate parameters of the model. Jang [1] shows that the ANFIS architecture can model non-linear functions, identify non-linear components in a control system, and predict the Mackey-Glass chaotic time series with remarkable results.

However, the hybrid learning procedure in ANFIS has several shortcomings. Firstly the hybrid algorithm requires the LSE to estimate linear parameters in the models. The LSE has following problems:

- The LSE relies on the calculation of the pseudo inverse $(A^T A)^{-1} A^T$ which in certain cases, the matrix $A^T A$ is ill-conditioned; this problem occurs when the measurements are only mildly related to the estimated parameters. In these cases, the LSE amplifies the measurement noise, and may be grossly inaccurate. This may occur even when the pseudo inverse itself can be accurately calculated numerically.

- The LSE seeks to minimize the norm of the measurement error $A\theta - f$. In many cases, one is truly interested in obtaining small error in the parameter θ, e.g., a small value of $\left\|\theta - \theta^*\right\|$. However, since θ is unknown, this quantity cannot be directly minimized.

- Better estimators can be constructed, an effect known as Stein's phenomenon [28, 29]. For example, if the measurement error is Gaussian, several estimators are known which dominate, or outperform, the least squares technique; the most common of these is the James-Stein estimator [28, 29].

Moreover ANFIS requires gradient decent back-propagation algorithm to estimate its non-linear parameters. This algorithm has following drawbacks:

- The algorithm is highly complex due to the fact that the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes, therefore for every node the gradient of the error with respect to ANFIS non-linear parameters needs to be calculated using the order derivative [4].

- It has been shown in [46] that by using the gradient decent back-propagation algorithm, there is a harmful and unforeseeable influence of the size of the partial derivative on the weight step.

All of these complexities eventually result in slow convergence of the parameters (the detailed discussions on the hybrid learning procedure and its shortcomings can be found in section 6.3).

Since ANFIS is an integrated system using the fuzzy inference system and adaptive networks hybrid learning procedures, this thesis will integrate the fuzzy inference system with a faster and more effective learning algorithm which is called the “Faster Adaptive Network Based Fuzzy Inference System” (FANFIS).

Like ANFIS, FANFIS is a universal approximator since it shares the same architecture as fuzzy inference systems which are universal approximator [49, 50].

Unlike ANFIS however, the FANFIS learning procedure does not require the LSE to estimate its linear parameters and the gradient decent back-propagation algorithm to estimate its non-linear parameters. The new procedure uses the same technique to estimate both linear and nonlinear parameters. In the FANFIS procedure the order derivatives are used only to determine the *direction* of the parameter step, but not the *size* of the change. It is possible to reduce the computational cost if the proposed procedure is used, due to the decrease in the number of iterations during the training process, as well as due to the fact that the algorithm allows substantial reduction in the amount of computational operations that have to be performed during each single iteration.

The results in this thesis show that FANFIS outperforms ANFIS and other competing models in predicting the chaotic time series [5], modeling a three-Input nonlinear function [36, 37, 38] and identification of dynamical systems [39]. We also used FANFIS to predict five major stock closing prices in New Zealand namely Air New Zealand Ltd. "A", Brierley Investments Ltd., Carter Holt Harvey Ltd., Lion Nathan Ltd. and Telecom Corporation of New Zealand Ltd.; and by using simple trading strategy, FANFIS outperforms other competing models in my previous research [40]. Thus we show that profitable forecasting is possible using FANFIS.

This thesis starts with the summary of fuzzy logic, fuzzy sets, membership functions and their operations in Chapter 2. These methods are essential to the development of fuzzy rules and fuzzy reasoning which will be explained in Chapter 3. By using fuzzy rules and fuzzy reasoning the fuzzy inference system is developed as explained in Chapter 4. In Chapter 5 the adaptive network which is an integrated part of ANFIS is introduced. Chapter 6 shows how fuzzy inference systems and adaptive networks integrate into ANFIS. Chapter 7 develops the new Faster ANFIS and its performance is shown to be superior to the existing systems. The conclusions of this thesis are given in Chapter 8.

Chapter 2

2 Fuzzy Logic

2.1 Introduction

L. Zadeh who published his first work on fuzzy sets in 1965 [6] introduced the theory of fuzzy logic. The basic idea of fuzzy logic is to allow not only the values 1 and 0, corresponding to *true* and *false* but the whole interval [0,1] as degrees of truth. This leads to a radical extension of classical logic. Zadeh was not the first one to introduce a multi-valued logic calculus. In the twenties, J. Lukasiewicz [7] had already developed multi-valued logic calculus but its application was limited until the introduction of computer technology in the late fifties.

Although critics continually state that all-important problems can be dealt with by classical means, use of fuzzy logic has become more wide spread. In particular, this is indicated by the success of fuzzy logic in the discipline of control techniques. In the beginning of the seventies Zadeh [8] introduced the concept of fuzzy logic control. In recent years it has been shown that, with special kind of fuzzy controllers, each continuous function on a compact set can be approximated to any degree of accuracy [14].

2.2 Fuzzy Sets

We obtain fuzzy sets by extending the membership predicate "$\in$" to the interval [0, 1] instead of only using the classical truth-values 0 and 1. This means that a set can contain points with a certain *degree*. This degree of membership can be considered in different ways. On the one hand the membership grade can be interpreted as a grade of probability;

on the other hand it can be regarded as a grade of possibility. In any case fuzzy set theory allows us to deal with data associated with uncertainty. This is an advantage we can use within neural networks.

Definition 2.1 *Fuzzy sets and membership functions*
If X is a collection of objects denoted generically by x, then a ***fuzzy set*** *A in X is defined as a set of ordered pairs:*

$$A = \left\{ \left(x, \mu_A(x) \right) \middle| x \in X \right\}$$

where $\mu_A(x)$ *is called the* ***membership function*** *(* ***MF****) for the fuzzy set A. The MF maps each element of X to a membership grade (or membership value) between 0 and 1.*

Example 2.1

Let $X = \mathbb{R}^+$ be the set of possible ages for human beings. Then the fuzzy set $A =$ "about 50 years old" may be expressed as

$$A = \left\{ \left(x, \mu_A(x) \right) \middle| x \in X \right\},$$

where

$$\mu_A(x) = \frac{1}{1 + \left(\frac{x - 50}{10} \right)^4}.$$

This is illustrated in Figure 2.1.

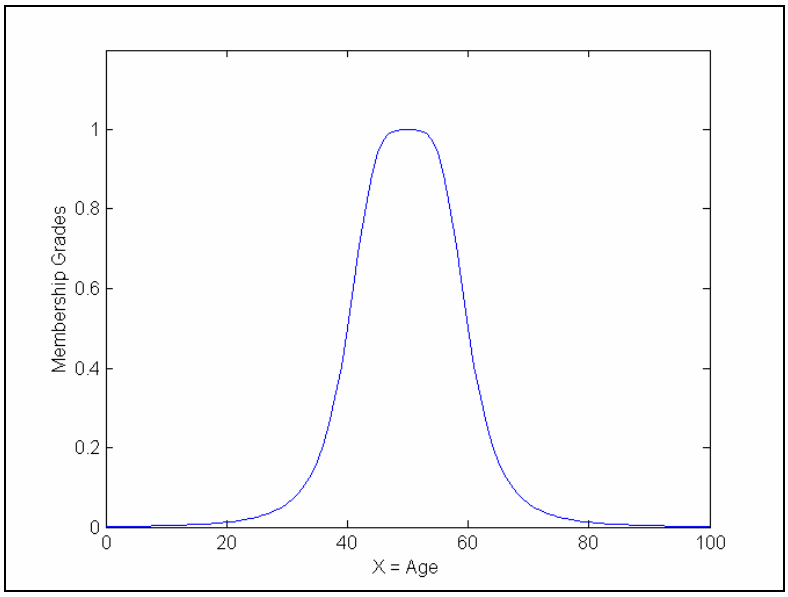


Figure 2.1 Fuzzy set A = "about 50 years old"

□

From the preceding examples, it is obvious that the construction of a fuzzy set depends on two things: the identification of a suitable universe of discourse and the specification of an appropriate membership function. The specification of membership functions is subjective, which means that the membership functions specified for the same concept (say, "sensible number of children in a family") by different persons may vary considerably. This subjectivity comes from individual differences in perceiving or expressing abstract concepts has little to do with randomness. Therefore, the subjectivity and non-randomness of a fuzzy set is the primary difference between the study of fuzzy sets and probability theory, which deal with objective treatment of random phenomena.

Obviously, the definition of a fuzzy set is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any values between 0 and 1. If the value of the membership function $\mu_A(x)$ is restricted to either 0 or 1, then A is

reduced to a classical set and $\mu_A(x)$ is the characteristic function of A. For clarity, we shall also refer to classical sets as ordinary sets, crisp sets, non-fuzzy sets, or just sets.

Usually X is referred to as the **universe of discourse,** or simply the **universe** and it may consist of discrete objects or continuous space.

For simplicity of notation, we now introduce an alternative way of denoting a fuzzy set. A fuzzy set *A* can be denoted as follows:

$$A = \begin{Bmatrix} \sum_{x_i \in X} \mu_A(x_i) \Big/ x_i & \text{if } X \text{ is a collection of discrete objects.} \\ \int_X \mu_A(x) \Big/ x & \text{if } X \text{ is a continuous space} \end{Bmatrix}$$

The summation and integration sign stands for the union of $(x, \mu_A(x))$ pairs; they do not indicate summation or integration in the usual sense. Similarly " $\Big/$ " is only a marker and does not imply division. For example the fuzzy set in example 2.1 can be rewritten as

$$A = \int_{\mathbb{R}^+} \frac{1}{1 + \left(\frac{x - 50}{10} \right)^4} \Big/ x.$$

2.3 MF Formulation and Parameterization

Since most fuzzy sets in use have a universe of discourse *X* consisting of the real line $\mathbb{R}$, it would be impractical to list all the pairs defining a membership function. A more convenient way to define an MF is to express it as a mathematical formula. Here are some of the MFs used in the fuzzy sets literature

Definition 2.2 *Triangular MFs*

*A **triangular MF** is specified by three parameters* $\{a,b,c\}$ *as follows:*

$$triangle(x;a,b,c)=\begin{cases}0, & x\le a.\\ \dfrac{x-a}{b-a}, & a\le x\le b.\\ \dfrac{c-x}{c-b}, & b\le x\le c.\\ 0, & c\le x.\end{cases}$$

The parameter $\{a,b,c\}$ *(with* $a<b<c$ *) determine the* x *coordinates of the three corners of the underlying triangular MF.*

Definition 2.3 *Trapezoidal MFs*

*A **trapezoidal MF** is specified by four parameters* $\{a,b,c,d\}$ *as follows:*

$$trapezoid\left(x;a,b,c,d\right)=\begin{cases}0, & x\le a.\\ \dfrac{x-a}{b-a}, & a\le x\le b.\\ 1, & b\le x\le c.\\ \dfrac{d-x}{d-c}, & c\le x\le d.\\ 0, & d\le x.\end{cases}$$

The parameters $\{a,b,c,d\}$ *with (* $a<b<c<d$ *) determine the x coordinates of the four corners of the underlying trapezoidal MF.*

Note that a trapezoidal MF with parameter $\{a,b,c,d\}$ reduces to a triangular MF when *b* is equal to *c*.

Due to their simple formulas and computational efficiency, both triangular and trapezoidal MFs have been used extensively, especially in real-time implementations. However, since the MFs are composed of straight-line segments, they are not smooth at the corner points specified by the parameters. In the following we introduce other types of MFs defined by smooth and nonlinear functions.

Definition 2.4 *Gaussian MFs*

*A **Gaussian MF** is specified by two parameters* $\{c,\sigma\}$*:*

$$gaussian(x;c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}.$$

A Gaussian MF is determined completely by *c* and σ; *c* represents the MFs center and σ determines the MF spread.

Definition 2.5 *Generalized bell MFs*

*A **generalized bell MF** (or **bell MF**) is specified by three parameters* $\{a,b,c\}$*:*

$$bell(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}},$$

where the parameter b is positive otherwise the shape of this MF becomes an upside-down bell.

Note that this MF is a direct generalization of the Cauchy distribution so it is also referred to as the **Cauchy MF**. A desired generalized bell MF can be obtained by a proper selection of the parameter set $\{a,b,c\}$. We can adjust *c* and *a* to vary the center and spread of the MF, and then use *b* to control the slopes at the cross-over points. Figure 2.2 shows the effects of changing each parameter.

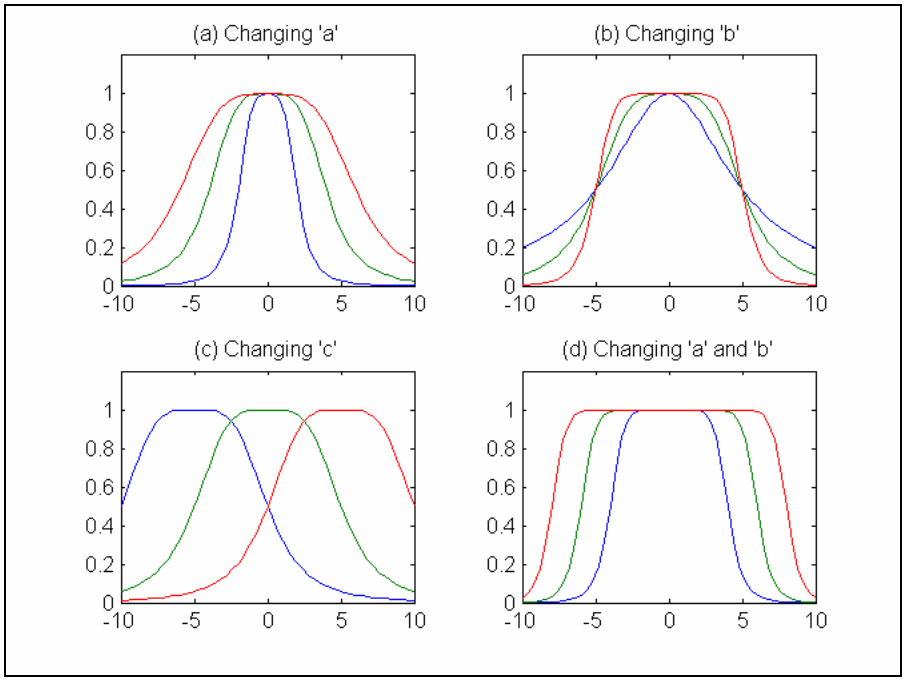


Figure 2.2. The effect of changing parameters in bell MFs.

Because of their smoothness and concise notation, Gaussian and bell MFs are becoming increasingly popular for specifying fuzzy sets.

2.3.1 MFs of Two dimensions

Sometimes it is advantageous or necessary to use MFs with two inputs, each in a different universe of discourse. MFs of this kind are generally referred to as two-dimensional MFs, whereas ordinary MFs (MFs with one input) are referred to as one-dimensional MFs. One natural way to extend one-dimensional MFs to two-dimensional ones is via cylindrical extension, defined next.

Definition 2.6 *Cylindrical extensions of one-dimensional fuzzy sets*

If A is a fuzzy set in X, then its ***cylindrical extension*** *in* $X \times Y$ *is a fuzzy set c(A) defined by*

$$c(A) = \int_{X \times Y} \mu_A(x)/(x,y).$$

The concept of cylindrical extension is quite straightforward; it is illustrated in Figure 2.3.

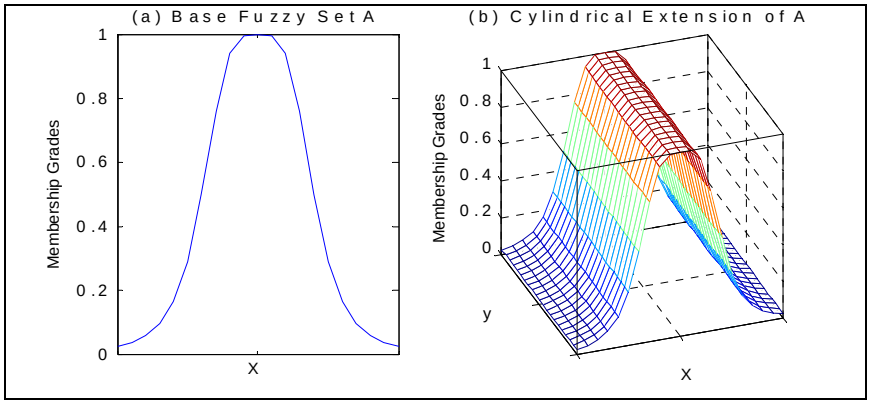


Figure 2.3. (a) Base set *A*; (b) its cylindrical extension *c(A)*

The operation of projection, on the other hand, decreases the dimension of a given (multidimensional) membership function.

Definition 2.7 *Projections of fuzzy sets*

Let R be a two-dimensional fuzzy set on $X \times Y$ *. Then the* ***projections*** *of R onto X and Y are defined as*

$$R_X = \int_X \left[\max_y \mu_R(x,y) \right] \Big/ x$$

and

$$R_Y = \int_Y \left[\max_x \mu_R(x,y) \right] \Big/ y,$$

respectively.

Figure 2.4(a) shows the F for fuzzy set *R*; Figure 2.4(b) and figure 2.4(c) are the projections of *R* onto *X* and *Y*, respectively.

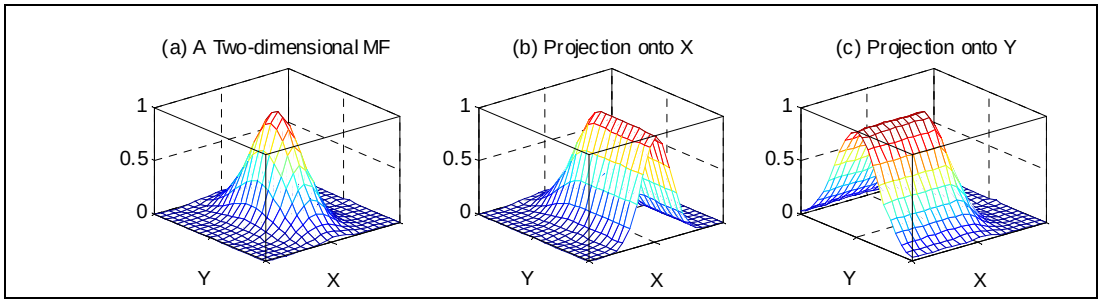


Figure 2.4. (a) Two dimensional fuzzy set *R*; (b) R_X **(projection of *R* onto *X*); and (c)** R_Y **(projection of *R* onto *Y*).**

2.4 Combining fuzzy sets

Fuzzy sets are combined in the application of fuzzy reasoning. The combination of fuzzy sets can be obtained using intersection (AND), union (OR) and complement (NOT) operations. The mathematical description of these and other operations for a set *x* and a subset *A* are given in definitions 2.6 to 2.9.

Definition 2.6 *Containment or subset*

Fuzzy set A is ***contained*** *in fuzzy set B (or, equivalently, A is a* ***subset*** *of B, or A is smaller than or equal to B) if and only if* $\mu_A(x) \le \mu_B(x)$ *for all x in X. In symbols,*

$$A \subseteq B \Leftrightarrow \mu_A(x) \le \mu_B(x) \forall x \in X.$$

Figure 2.5 illustrates the concept of $A \subseteq B$

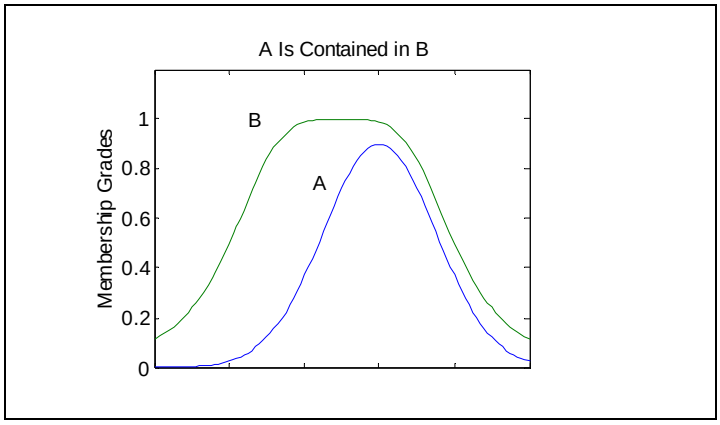


Figure 2.5 The concept of $A \subseteq B$

Definition 2.7 *Union (disjunction)*

*The **union** of two fuzzy sets A and B is a fuzzy set C, written as* $C = A \cup B$ *or C = A OR B, whose MF is related to those of A and B by*

$$\mu_C(x) = \max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \vee \mu_B(x) \forall x \in X.$$

A more intuitive but equivalent definition of union is the "smallest" fuzzy set containing both *A* and *B.* Alternatively, if *D* is any fuzzy set that contains both *A* and *B,* then it also contains $A \cup B$.

Definition 2.8 *Intersection (conjunction)*

*The **intersection** of two fuzzy sets A and B is a fuzzy set C, written as* $C = A \cap B$ *or C = A AND B, whose MF is related to those of A and B by*

$$\mu_C(x) = \min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \wedge \mu_B(x) \forall x \in X.$$

As in the case of the union, it is obvious that the intersection of *A* and *B* is the "largest" fuzzy set which is contained in both *A* and *B.* This reduces to the ordinary intersection operation if both *A* and *B* are non-fuzzy.

Definition 2.9 *Complement (negation)*

*The **complement** of fuzzy set A, denoted by* $\overline{A}$ *(*$\neg A$*, NOT A), is defined as*

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \forall x \in X.$$

Figure 2.6(a) illustrates two fuzzy sets *A* and *B*; Figure 2.6(b) is the compliment of A; Figure 2.6(c) is the union of *A* and *B*; and Figure 2.6(d) is the intersection of *A* and *B*.

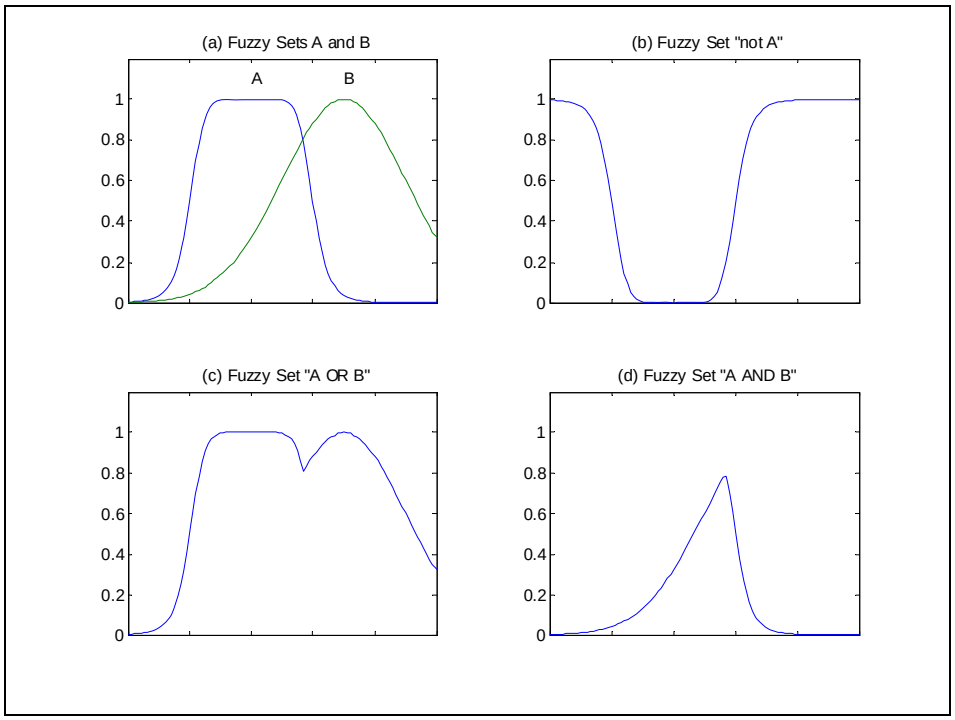


Figure 2.6. (a) two fuzzy sets A and B; (b) $\overline{A}$**; (c)** $A \cup B$**; (d)** $A \cap B$**.**

These fuzzy set operations perform exactly as the corresponding operations for ordinary sets if the values of the membership functions are restricted to either 0 or 1. However it is understood that these functions are not the only possible generalizations of the crisp set

operations. For each of the operations above, several different classes of functions with desirable properties have been proposed subsequently in the literature; we will introduce some of these functions in Section 2.4.1.

Next we define other operations on fuzzy sets, which are also direct generalization of operations on ordinary sets.

Definition 2.10 *Cartesian product and co-product*

Let A and B be fuzzy sets in X and Y, respectively. The ***Cartesian product*** *of A and B, denoted by* $A \times B$*, is a fuzzy set in the product space* $X \times Y$ *with the membership function*

$$\mu_{A\times B}(x,y) = \min\left(\mu_A(x), \mu_B(y)\right).$$

Similarly, the ***Cartesian co-product*** $A + B$ *is a fuzzy set with the membership function*

$$\mu_{A+B}(x,y) = \max\left(\mu_A(x), \mu_B(y)\right).$$

Although these classical fuzzy set operators possess more rigorous axiomatic properties, they are not the only ways to define reasonable and consistent operations on fuzzy sets. The following section examines other viable definitions of the fuzzy intersection and union operator.

2.4.1 Generalized Fuzzy Intersection and Union Operation

The intersection of two fuzzy sets A and B is specified in general by a function $T:[0,1]\times[0,1]\to[0,1]$, which aggregates two membership grades as follows:

$$\mu_{A\cap B}(x)=T\left(\mu_A(x),\mu_B(x)\right)=\mu_A(x)\,\tilde{*}\,\mu_B(x),$$

where $\tilde{*}$ is a binary operator denoting the function *T*. This class of fuzzy intersection operators, which are usually referred to as T-norm (triangular norm) operators, meets the following basic requirements.

Definition 2.11 *T-norm operators*

*A **T-norm operator** is a two-place function $T(\cdot,\cdot)$ satisfying*

$T(0,0)=0,\ T(a,1)=T(1,a)=a$	(boundary)
$T(a,b)\le T(c,d)$ if $a\le c$ and $b\le d$	(monotonicity)
$T(a,b)=T(b,a)$	(commutativity)
$T(a,T(b,c))=T(T(a,b),c)$	(associativity)

The first requirement imposes the correct generalization to crisp sets. The second requirement implies that a decrease in the membership values in A or B cannot produce an increase in the membership value in $A\cap B$. The third requirement indicates that the operator is indifferent to the order of the fuzzy sets to be combined. Finally, the fourth requirement allows us to take the intersection of any number of sets in any order of pair-wise groupings. The following example illustrates four of the most frequently encountered T-norm operators.

Minimum: $T_{\min}(a,b) = \min(a,b) = a \wedge b.$

Algebraic product: $T_{ap}(a,b) = ab.$

Bounded product: $T_{bp}(a,b) = 0 \vee (a+b-1).$

Drastic product: $T_{dp}(a,b) = \begin{cases} a, & \text{if } b = 1. \\ b, & \text{if } a = 1. \\ 0, & \text{if } a,b < 1. \end{cases}$

With the understanding that a and b are between 0 and 1, we can draw surface plots of these four T-norm operators as functions of a and b; see the first row of Figures 2.7.1 and 2.7.2. The second row of Figures 2.71 and 2.7.2 shows the corresponding surfaces when $a = \mu_A(x) = \text{trapezoid}(x;3,8,12,17)$ and $b = \mu_B(y) = \text{trapezoid}(x;3,8,12,17)$; these two-dimensional MFs can be viewed as the Cartesian product of A and B under four different T-norm operators. From Figures 2.7.1 and 2.7.2, it can be observed that $T_{dp}(a,b) \le T_{bp}(a,b) \le T_{ap}(a,b) \le T_{\min}(a,b)$.

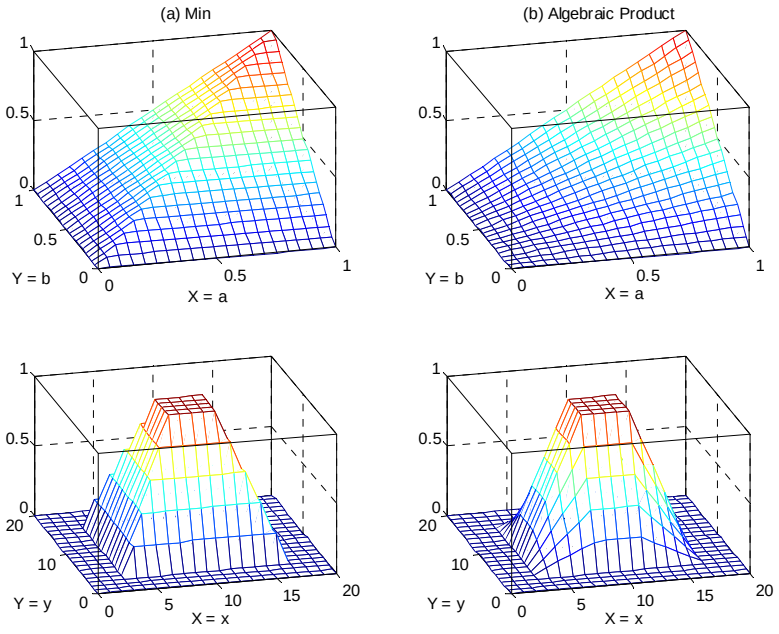


Figure 2.7.1. (1st row) $T_{\min}(a,b)$ and $T_{ap}(a,b)$, (2nd row) The corresponding surfaces for $a = \text{trapezoid}(x,3,8,12,17)$ and $b = \text{trapezoid}(x,3,8,12,17)$.

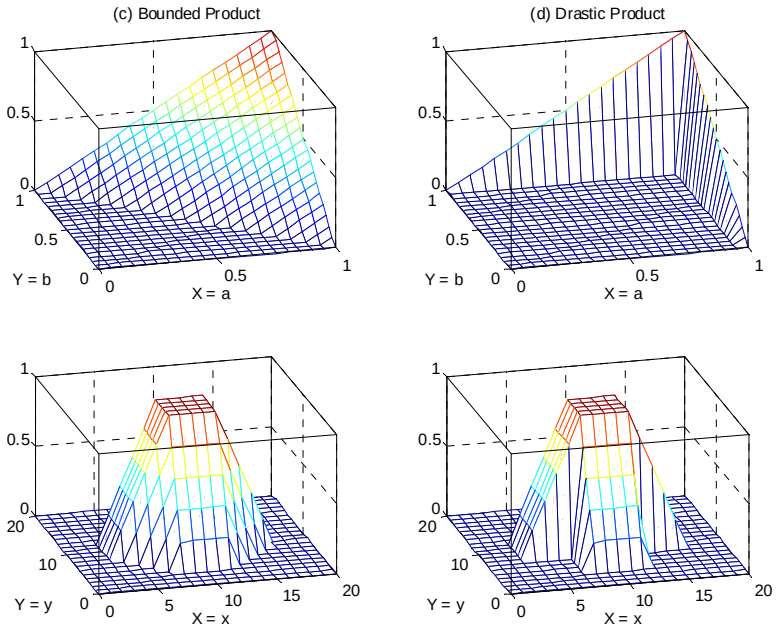


Figure 2.7.2. (1st row) $T_{bp}(a,b)$ **and** $T_{dp}(a,b)$**; (2nd row) The corresponding surfaces for** $a =$ **trapezoid**$(x,3,8,12,17)$ **and** $b =$ **trapezoid**$(x,3,8,12,17)$

Like fuzzy intersection, the fuzzy union operator is specified in general by a function $S:[0,1]\times[0,1]\to[0,1]$. In symbols,

$$\mu_{A\cup B}(x) = S(\mu_A(x),\mu_B(x)) = \mu_A(x)\,\tilde{+}\,\mu_B(x),$$

where $\tilde{+}$ is a binary operator for the function S. This class of fuzzy union operators, which are often referred to as T-conorm operators, satisfy the following basic requirements.

Definition 2.12 *T-conorm operators*

*A **T-conorm operator** is a two place function* $S(\cdot,\cdot)$ *satisfying*

$$
\begin{aligned}
&S(1,1)=1,\ S(0,a)=S(a,0)=a && \text{(boundary)}\\
&S(a,b)\le S(c,d)\ \text{if } a\le c \text{ and } b\le d && \text{(monotonicity)}\\
&S(a,b)=S(b,a) && \text{(commutativity)}\\
&S(a,S(b,c))=S(S(a,b),c) && \text{(associativity)}
\end{aligned}
$$

The justification of these basic requirements is similar to that of the requirements for T-norm operators.

Corresponding to the four T-norm operators in the previous example, we have the following four T-conorm operators.

$$
\begin{aligned}
&\text{Maximum:} && S_{\max}(a,b)=\max(a,b)=a\vee b.\\
&\text{Algebraic sum:} && S_{as}(a,b)=a+b-ab.\\
&\text{Bounded sum:} && S_{bs}(a,b)=1\wedge(a+b).\\
&\text{Drastic sum:} && S_{ds}(a,b)=\begin{cases} a, & \text{if } b=0.\\ b, & \text{if } a=0.\\ 0, & \text{if } a,b>0. \end{cases}
\end{aligned}
$$

The first row of Figures 2.8.1 and 2.8.2 shows the surface plots of these four T-conorm operators. The second row demonstrates the corresponding two-dimensional MFS when $a=\mu_A(x)=\text{trapezoid}(x;3,8,12,17)$ and $b=\mu_B(y)=\text{trapezoid}(x;3,8,12,17)$; these two-dimensional MFs can be viewed as the Cartesian co-product of *A* and *B* under four different T-norm operators.

It can also be observed that $S_{\max}(a,b) \le S_{as}(a,b) \le S_{bs}(a,b) \le S_{ds}(a,b)$.

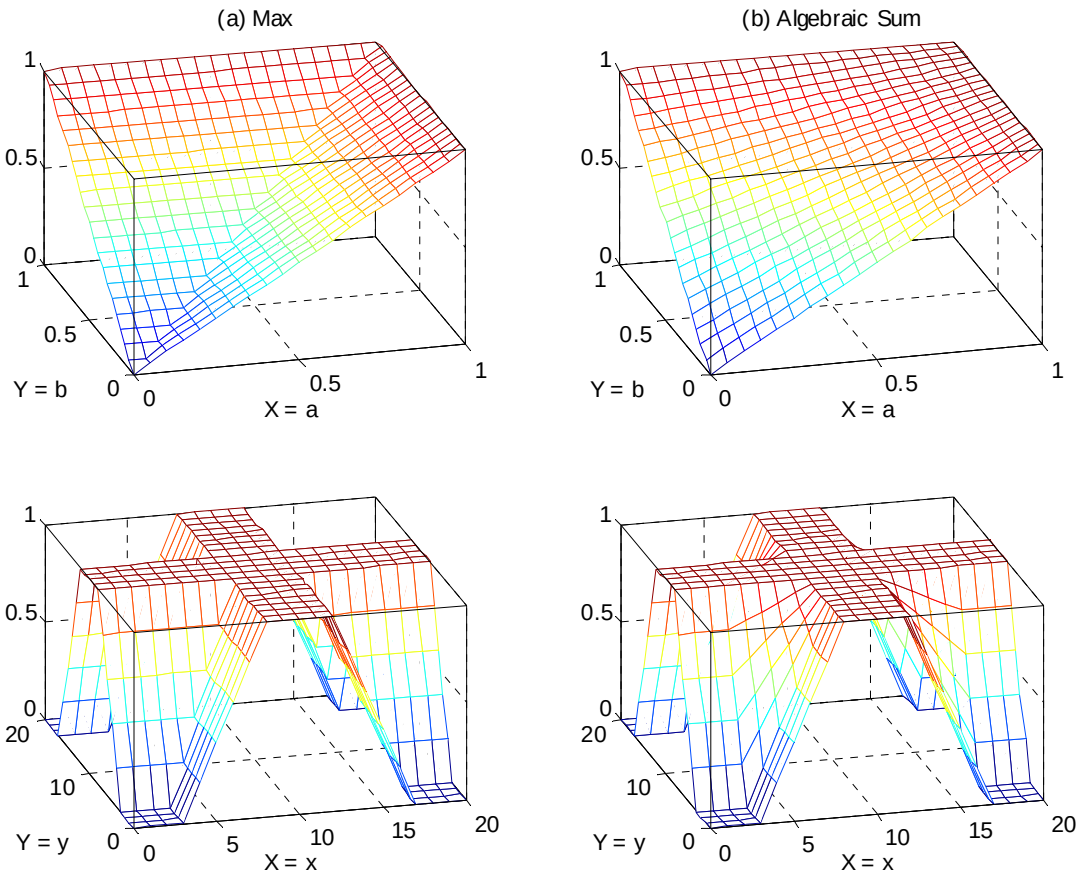


Figure 2.8.1 (1st row) $S_{\max}(a,b)$ **and** $S_{as}(a,b)$**, (2nd row) the corresponding surfaces for** $a =$ **trapezoid**$(x,3,8,12,17)$ **and** $b =$ **trapezoid**$(x,3,8,12,17)$

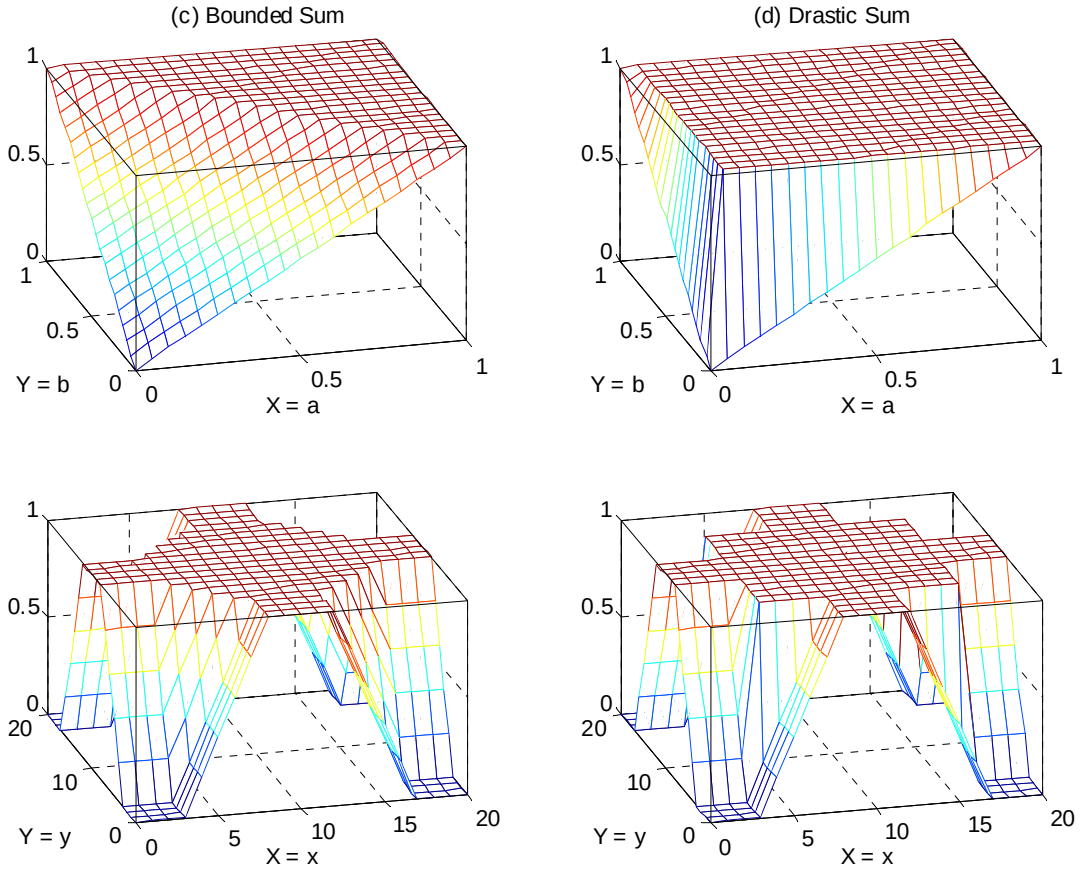


Figure 2.8.2 (1st row) $S_{bs}(a,b)$ **and** $S_{ds}(a,b)$**; (2nd row) the corresponding surfaces for** $a =$ **trapezoid**$(x,3,8,12,17)$ **and** $b =$ **trapezoid**$(x,3,8,12,17)$

Chapter 3

3 Fuzzy Rules and Fuzzy Reasoning

3.1 Introduction

Fuzzy rules and fuzzy reasoning are the backbone of fuzzy inference systems, which are the most important modelling tool based on fuzzy set theory. They have been successfully applied to a wide range of areas, such as automatic control, expert systems, pattern recognition, time series prediction, and classification.

We shall start by giving definitions and examples of the extension principle and fuzzy relations, which are the rationales behind fuzzy reasoning.

3.2.1 Extension Principle

The **extension principle** [6, 9] is a basic concept of fuzzy set theory that provides a general procedure for extending crisp domains of mathematical expressions to fuzzy domains. This procedure generalizes a common point-to-point mapping of a function $f(\cdot)$ to a mapping between fuzzy sets. More specifically, suppose that f is a function from *X* to *Y* and *A* is a fuzzy set on *X* defined as

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n.$$

Then the extension principle states that the image of fuzzy set A under the mapping $f(\cdot)$ can be expressed as a fuzzy set B,

$$B = f(A) = \mu_A(x_1)/y_1 + \mu_A(x_2)/y_2 + \ldots + \mu_A(x_n)/y_n,$$

where $y_i = f(x_i),\ i = 1,\ldots,n.$ In other words, the fuzzy set *B* can be defined through the values of $f(\cdot)$ in $x_1,\ldots,x_n$. If $f(\cdot)$ is a many-to-one mapping, then there exists $x_1, x_2 \in X,\ x_1 \neq x_2,$ such that $f(x_1) = f(x_2) = y^*, y^* \in Y.$ In this case, the membership grade of *B* at $y = y^*$ is the maximum of the membership grades of *A* at $x = x_1$ and $x = x_2$, since $f(x) = y^*$ may result from either $x = x_1$ or $x = x_2$.

More generally, we have

$$\mu_B(y) = \max_{x = f^{-1}(y)} \mu_A(x).$$

A simple example follows

Example 3.1 *Application of the extension principle to fuzzy sets*

Let $\mu_A(x) = bell(x; 1.5, 2, 0.5)$ and $f(x) = \begin{cases} (x-1)^2 - 1, & \text{if } x \geq 0. \\ x, & \text{if } x \leq 0. \end{cases}$

Figure 3.1(a) is a plot of $y = f(x)$; Figure 3.1(c) is $\mu_A(x)$, the MF of *A*. After employing the extension principle, we obtain a fuzzy set *B*; its MF is shown in Figure 3.1(b), where the plot of $\mu_B(y)$ is rotated 90 degree for easy viewing. Since $f(x)$ is a many to one mapping for $x \in [-1,\ 2],$ the max operator is used to obtain the membership grades of *B* when $y \in [0,\ 1].$ This causes discontinuities of $\mu_B(y)$ at $y = 0$ and -1.

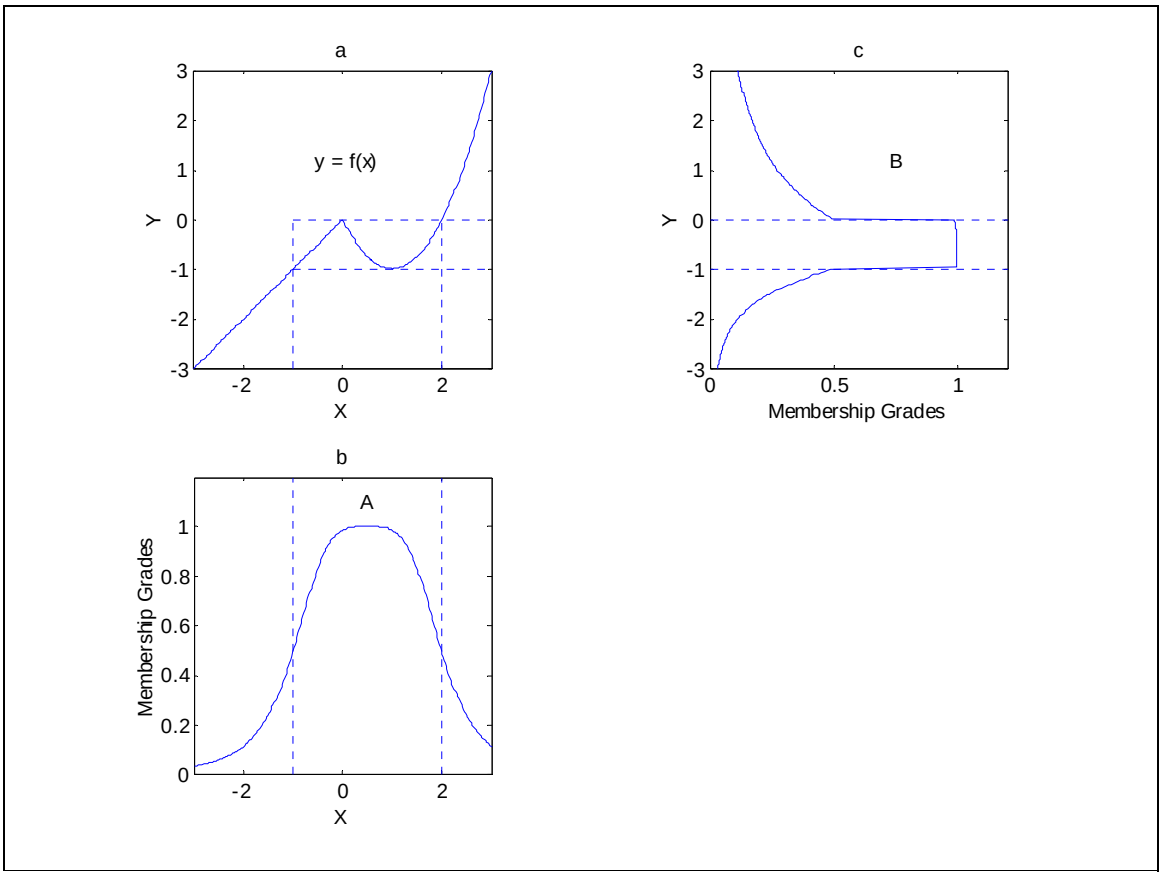


Figure 3.1 Extension principle on fuzzy sets as explained in Example 3.1

□

Now we consider a more general situation. Suppose that *f* is a mapping from *n*-dimensional product space $X_1 \times \cdots X_n$ to a single universe *Y* such that $f(x_1,\ldots,x_n) = y$, and there is a fuzzy set A_i in each X_i $i = 1,\ldots,n$.

Since each element in an input vector $(x_1,\ldots,x_n)$ occurs *simultaneously*, this implies an AND operation. Therefore, the membership grade of fuzzy set *B* induced by the mapping *f* should be the minimum of the membership grades of the constituent fuzzy set A_i, $i = 1,\ldots,n$. With this understanding, we give a complete formal definition of the extension principle.

Definition 3.1 *Extension principle*

Suppose that function f is a mapping from an n-dimensional Cartesian product space $X_1 \times X_2 \times \cdots X_n$ *to a one-dimensional universe Y such that* $y = f(x_1, \ldots, x_n)$, *and suppose* $A_1, \ldots, A_n$ *are n fuzzy sets in* $X_1, \ldots, X_n$, *respectively. Then the extension principle asserts that the fuzzy set B induced by the mapping f is defined by*

$$\mu_B(y) = \begin{cases} \sup_{(x_1, \ldots x_n) \in f^{-1}(y)} \min\left\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\right\}, & f^{-1}(y) \neq 0 \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad , & \textit{otherwise} \end{cases}$$

The foregoing extension principle assumes that $y = f(x_1, \ldots, x_n)$ is a crisp function. In cases where *f* is a fuzzy function [or, more precisely, when $y = f(x_1, \ldots, x_n)$ is a fuzzy set characterized by an (*n* + l)-dimensional MF], then we can employ the compositional rule of inference introduced in the following section to find the induced fuzzy set *B*.

3.1.2 Fuzzy Relation

Binary fuzzy relations are fuzzy sets in $X \times Y$ which map each element in $X \times Y$ to a membership grade between 0 and 1. In particular, unitary fuzzy relations are fuzzy sets with one-dimensional membership functions; binary fuzzy relations are fuzzy sets with two dimensional membership functions.

Definition 3.2 *Binary fuzzy relation*

Let X and Y be two universe of discourse. Then

$$R = \left\{\left((x, y), \mu_R(x, y)\right) \mid (x, y) \in X \times Y\right\}$$

*is a **binary fuzzy relation** in* $X \times Y$

Example 3.2 *Binary fuzzy relation*

Let $X = Y = \mathbb{R}^+$ (the positive real line) and R = "y is much greater than x." The MF of the fuzzy relation R can be subjectively defined as

$$\mu_R(x, y) = \begin{cases} \dfrac{y - x}{x + y + 2}, & \text{if } y > x. \\ 0, & \text{if } y \le x. \end{cases}$$

If X = {3, 4, 5} and Y = {3, 4, 5, 6, 7}, then it is convenient to express the fuzzy relation R as a **relation matrix:**

$$R = \begin{bmatrix} 0 & 0.111 & 0.200 & 0.273 & 0.333 \\ 0 & 0 & 0.091 & 0.167 & 0.231 \\ 0 & 0 & 0 & 0.077 & 0.143 \end{bmatrix},$$

where the element at row i and column j is equal to the membership grade between the i^{th} element of X and j element of Y.

Other common examples of binary fuzzy relations are as follows:

- x is close to y (x and y are numbers)
- x depends on y (x and y are events)
- x and y look alike (x and y are persons, objects, and so on)
- If x is large, then y is small (x is an observed reading and y is a corresponding action).

The last expression, "If x is A, then y is B," is used repeatedly in a fuzzy inference system. We will explore fuzzy relations of this kind in section 3.3.

□

Fuzzy relations in different product spaces can be combined through a composition operation. Different composition operations have been suggested for fuzzy relations; the best known is the max-min composition proposed by Zadeh [6].

Definition 3.3 *Max-min composition*

Let R_1 *and* R_2 *be two fuzzy relations defined on* $X \times Y$ *and* $Y \times Z$ *, respectively. The* ***max-min composition*** *of* R_1 *and* R_2 *is a fuzzy set defined by*

$$R_1 \circ R_2 = \left\{ \left[(x,z), \max_y \min\left(\mu_{R_1}(x,y), \mu_{R_2}(y,z) \right) \right] \middle| x \in X, y \in Y, z \in Z \right\},$$

or equivalently,

$$\begin{aligned} \mu_{R_1 \circ R_2}(x,z) &= \max_y \min\left[\mu_{R_1}(x,y), \mu_{R_2}(y,z) \right] \\ &= \vee_y \left[\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z) \right] \end{aligned}$$

When R_1 and R_2 are expressed as relation matrices, the calculation of $R_1 \circ R_2$ is almost the same as matrix multiplication, except that $\times$ and + are replaced by "max" and "min", respectively. For this reason, the max-min composition is also called the **max-min product.**

3.3 Fuzzy If-Then Rules

A fuzzy if-then rule (also known as fuzzy rule, fuzzy implication, or fuzzy conditional statement) assumes the form

if x is A then y is B,

where *A* and *B* are linguistic values defined by fuzzy sets on universes of discourse *X* and *Y,* respectively. Often "*x* is *A*" is called the **antecedent or premise,** while "*y* is *B*" is called the **consequence or conclusion.** Examples of fuzzy if-then rules **are** widespread in our daily linguistic expressions, such as the following:

• If pressure is high, then volume is small.

• If the road is slippery, then driving is dangerous.

• If a tomato is red, then it is ripe.

• If the speed is high, then apply the brake a little.

Before we can employ fuzzy if-then rules to model and analyze a system, first we have to formalize what is meant by the expression "if *x* is *A* then *y* is *B*", which is sometimes abbreviated as $A \to B$. In essence, the expression describes a relation between two variables *x* and *y;* this suggests that a fuzzy if-then rule be defined as a binary fuzzy relation *R* on the product space $X \times Y$ generally speaking, there are two ways to interpret the fuzzy rule $A \to B$. If we interpret $A \to B$ as *A* **coupled with** *B*, then

$$R = A \to B = A \times B = \int_{X \times Y} \mu_A(x) \tilde{*} \mu_B(y) \Big/ (x, y),$$

where $\tilde{*}$ is a T-norm operator and $A \to B$ is used again to represent the fuzzy relation R. On the other hand, if $A \to B$ is interpreted as *A* **entails** *B,* then it can be written as four different formulas:

Material implication:

$$R = A \to B = \neg A \cup B.$$

Propositional calculus:

$$R = A \to B = \neg A \cup (A \cap B).$$

Extended propositional calculus:

$$R = A \to B = (\neg A \cap \neg B) \cup B.$$

Generalization of modus ponens:

$$\mu_R(x, y) = \sup\{c \mid \mu_A(x) \tilde{*} c \le \mu_B(y) \text{ and } 0 \le c \le 1\}$$

Although these four formulas are different in appearance, they all reduce to the familiar identity $A \to B \equiv \neg A \cup B$ when *A* and *B* are propositions in the sense of two-valued logic.

Based on these two interpretations and various T-norm and T-conorm operators, a number of qualified methods can be formulated to calculate the fuzzy relation $R = A \to B$. Note that *R* can be viewed as a fuzzy set with a two-dimensional MF

$$\mu_R(x, y) = f(\mu_A(x), \mu_B(y)) = f(a, b),$$

with $a = \mu_A(x)$, $b = \mu_B(y)$, where the function f, called the **fuzzy implication function,** performs the task of transforming the membership grades of *x* in *A* and *y* in *B* into those of (x, y) in $A \to B$.

Suppose that we adopt the first interpretation, "A coupled with B," as the meaning of $A \to B$. Then four different fuzzy relations $A \to B$ result from employing four of the most commonly used T-norm operators.

1. $R_m = A \times B = f_m(a,b) = a \wedge b$. This relation, which was by Mamdani [11], results from using the min operator for conjunction.

2. $R_p = A \times B = f_p(a,b) = ab$. Proposed by Larsen [12], this relation is based on using the algebraic product operator for conjunction.

3. $R_{bp} = A \times B = 0 \vee (a + b - 1)$. This formula employs the bounded product operator for conjunction.

4. $R_{dp} = A \times B = f(a,b) = a \hat{\cdot} b = \begin{cases} a & \text{if } b = 1. \\ b & \text{if } a = 1. \\ 0 & \text{otherwise.} \end{cases}$

This formula uses the drastic product operator for conjunction.

The first row of Figures 3.2.1 and 3.2.2 shows these four fuzzy implication functions [with $a = \mu_A(x)$ and $b = \mu_B(y)$]; the second row shows the corresponding fuzzy relations R_m, R_p, R_{bp} and R_{dp} when $\mu_A(x)$ = bell(x; 4, 3, 10) and $\mu_B(y)$ = bell(y; 4, 3, 10).

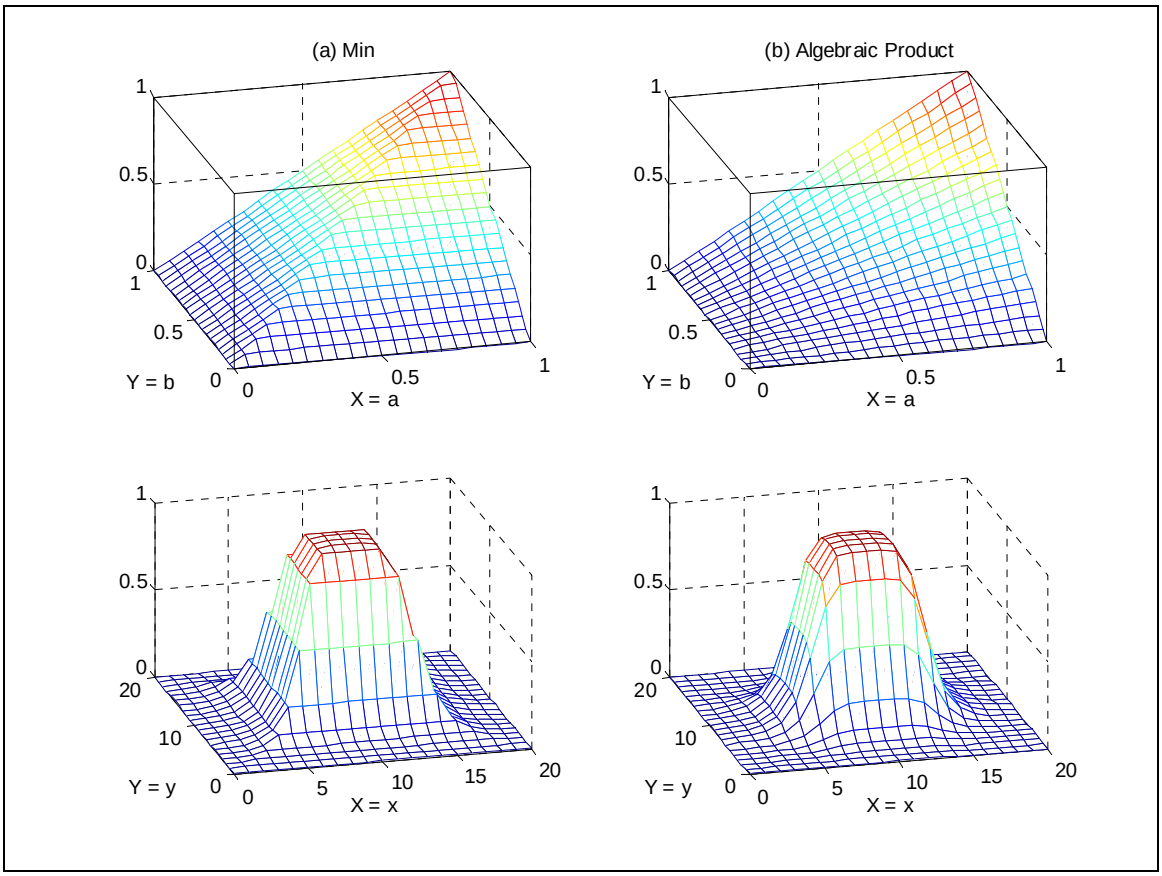


Figure 3.2.1 Fist row: Fuzzy implication functions R_m and R_p ; second row: The corresponding fuzzy relation

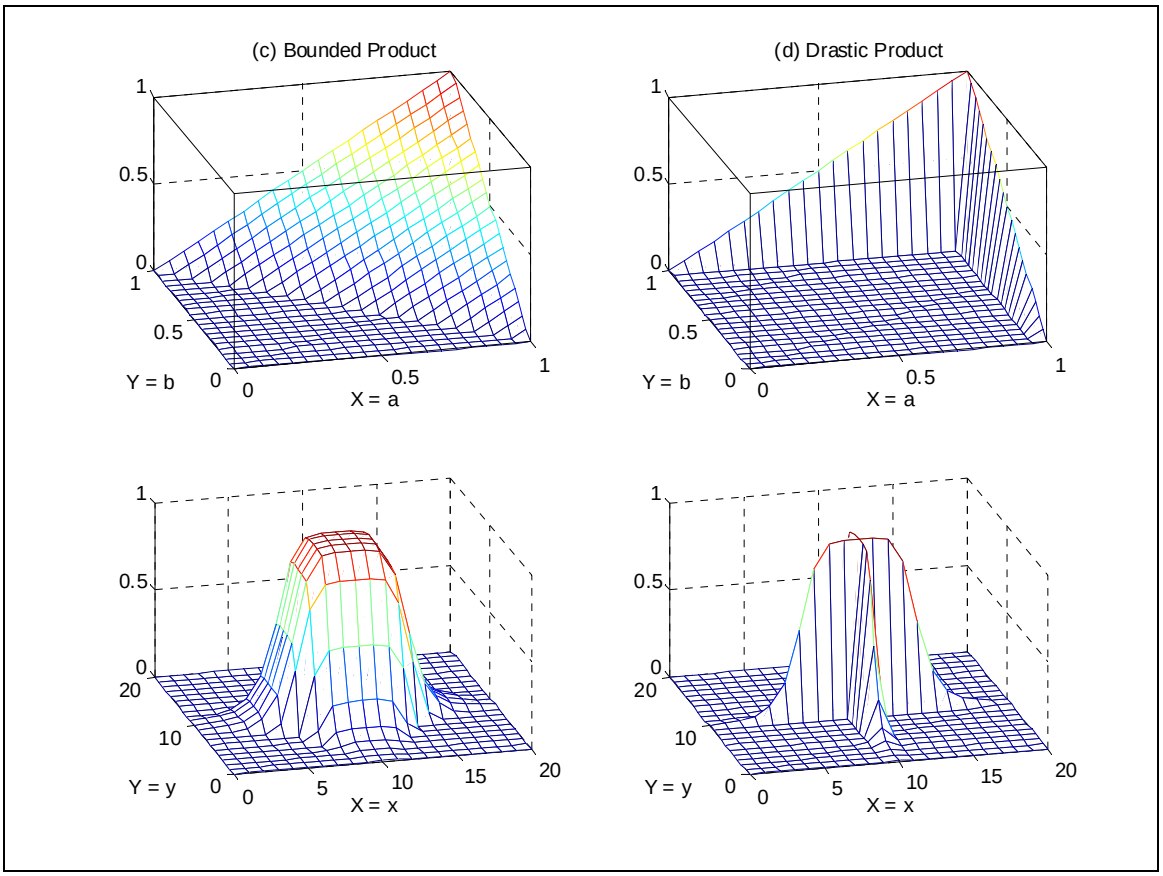


Figure 3.2.2 First row: Fuzzy implications function R_{bp} and R_{dp} ; second row: The corresponding fuzzy relation

When we adopt the second interpretation, "*A* entails B," as the meaning of $A \to B$, again there are a number of fuzzy implication functions that are reasonable candidates. The following four have been proposed in the literature:

1. $R_a = \neg A \cup B = f_a(a,b) = 1 \wedge (1 - a + b)$. This is Zadeh's arithmetic rule, which using the bounded sum operator for $\cup$.

2. $R_{mm} = \neg A \cup (A \cap B) = f_{mm}(a,b) = (1-a) \vee (a \wedge b)$. This is Zadeh's max-min rule, which using min for $\cap$ and max for $\cup$.

3. $R_s = \neg A \cup B = f_s(a,b) = (1-a) \vee b$. This is Boolean fuzzy implication using max for $\cup$.

4. $R_\Delta = f_\Delta(a,b) = a \tilde{<} b = \begin{cases} 1 & \text{if } a \le b. \\ b/a & \text{if } a > b. \end{cases}$

This is Goguen's fuzzy implication, using the algebraic product for the T-norm operator. Figures 3.3.1 and 3.3.2 shows these four fuzzy implication functions [with $a = \mu_A(x)$ and $b = \mu_B(y)$]; and the resulting fuzzy relations R_a, R_{mm}, R_s and R_Δ when $\mu_A(x)$ = bell(x; 4, 3, 10) and $\mu_B(y)$ = bell(y; 4, 3, 10).

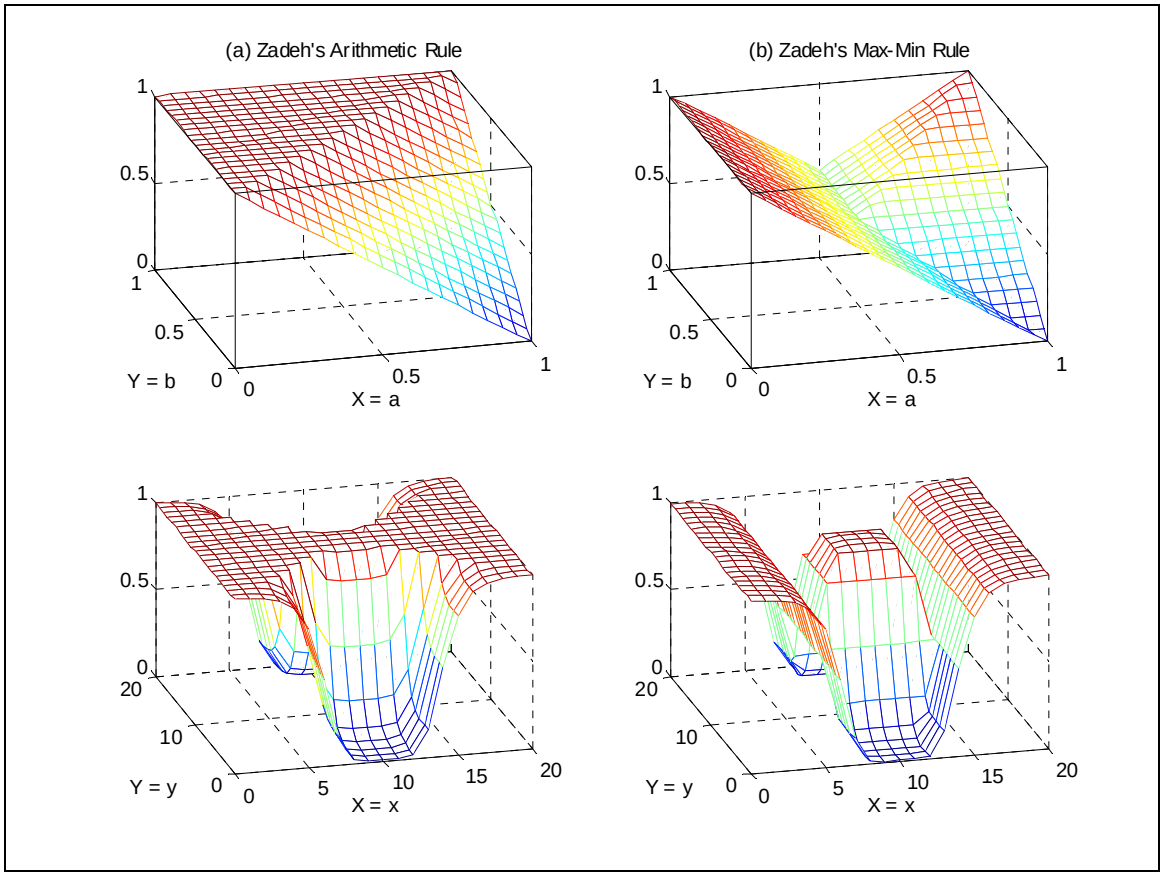


Figure 3.3.1 First row: fuzzy implication function R_a and R_{mm}; second row: the corresponding fuzzy relation

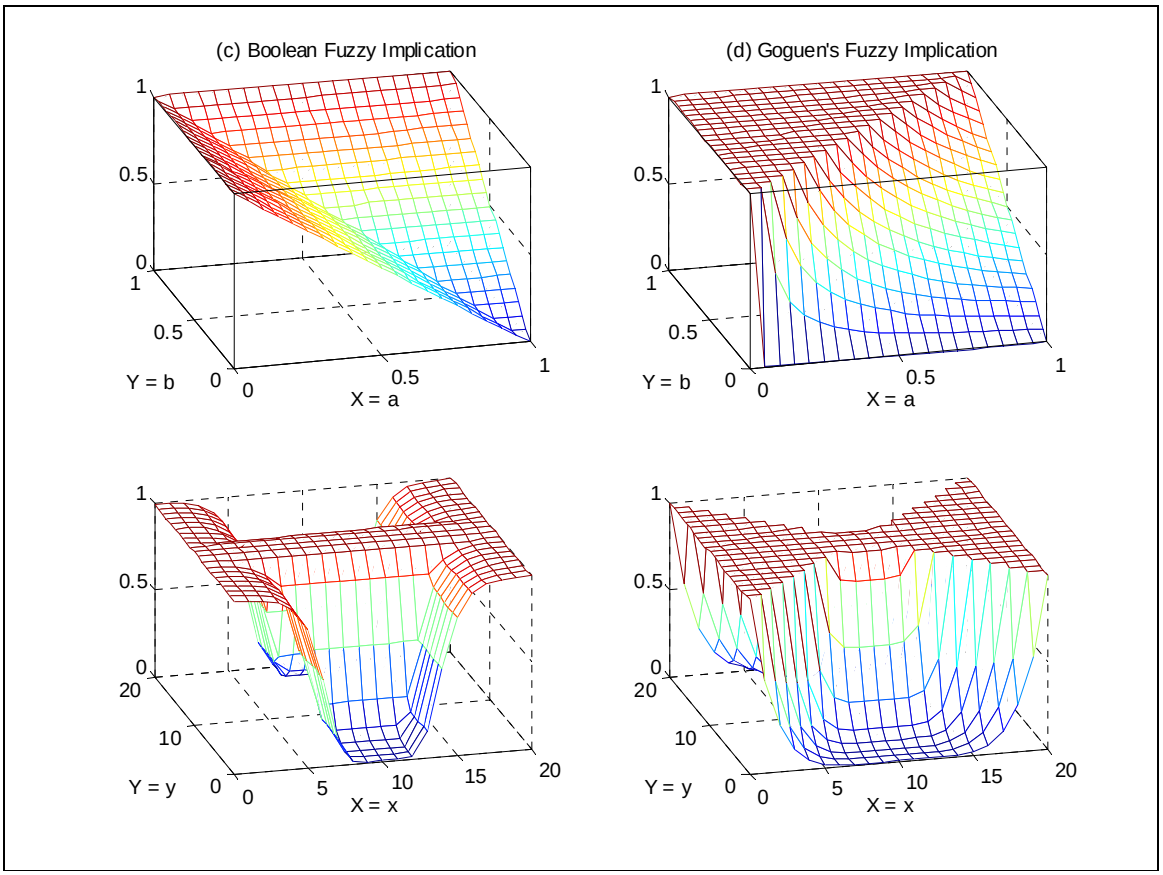


Figure 3.3.2 First row: fuzzy implication function R_s and R_Δ ; second row: the corresponding fuzzy relation.

It should be kept in mind that the fuzzy implication functions introduced here are by no means exhaustive. Interested readers can find other feasible fuzzy implication functions in [12].

3.4 Fuzzy Reasoning

Fuzzy reasoning, also known as approximate reasoning, is an inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts. Before introducing fuzzy reasoning, we shall discuss the compositional rule of inference, which plays a key role in fuzzy reasoning.

3.4.1 Compositional Rule of Inference

The compositional rule of inference proposed by Zadeh [10] is a generalization of the extension principle. Suppose that we have a curve $y = f(x)$ that regulates the relation between x and y. When we are given $x = a$, then from $y = f(x)$ we can infer that $y = b = f(a)$. A generalization of the aforementioned process would allow a to be an interval and $f(x)$ to be an interval-valued function.

To find the resulting interval $y = b$ corresponding to the interval $x = a$, we first construct a cylindrical extension of a and then find its intersection I with the interval-valued curve. The projection of I onto the *y-axis* yields the interval $y = b$.

Going one step further in our generalization, we assume that F is a fuzzy relation *on* $X \times Y$ and A is a fuzzy set of X as shown in Figures 3.4(a) and 3.4(b). To find the resulting fuzzy set B, again we construct a cylindrical extension $c(A)$ with base A. The intersection of $c(A)$ and F [Figure 3.4(c)] forms the analogue of the region of intersection I. By projecting $c(A) \cap F$ onto the y-axis, we infer *y as a fuzzy set B* on the *y-axis*, as shown in Figure 3.4(d).

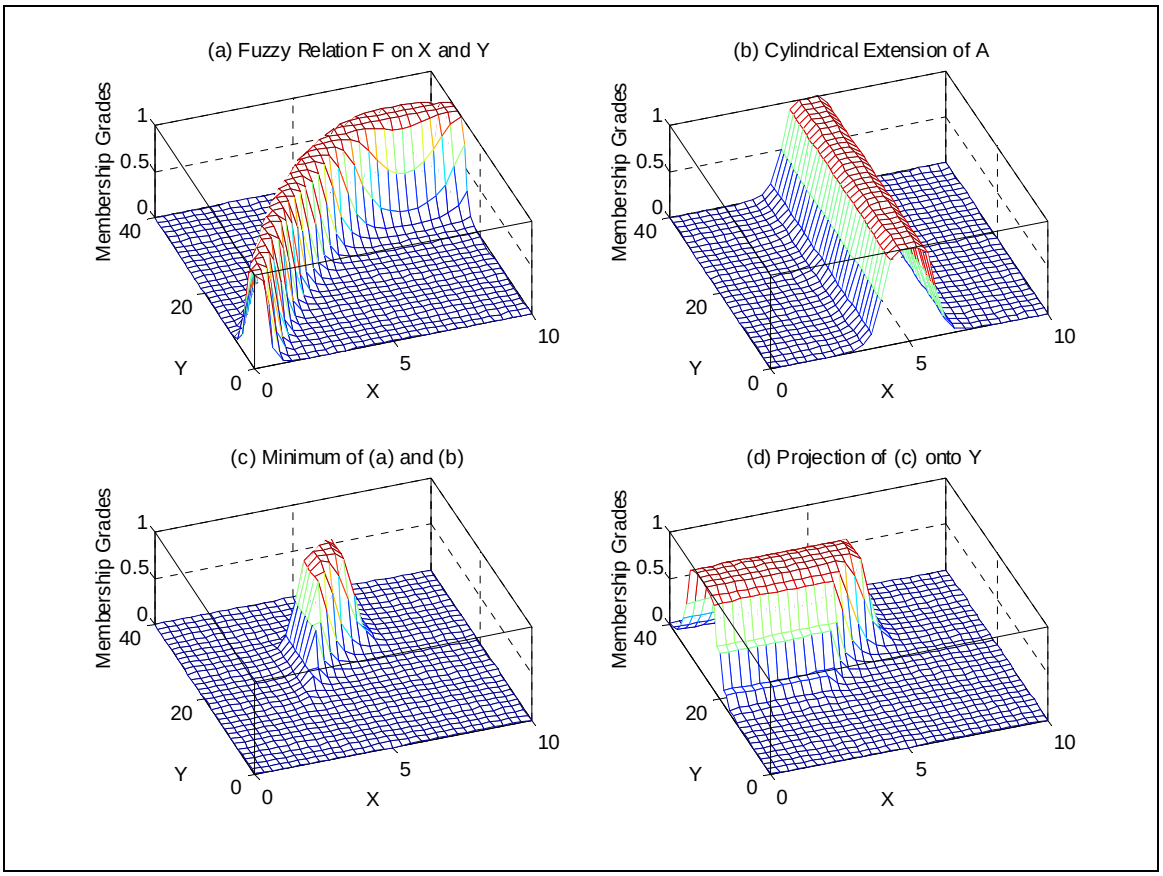


Figure 3.4 Compositional rule of inference.

Specifically, let $\mu_A, \mu_{c(A)}, \mu_B,$ and μ_F be the MFs of *A*, *c(A)*, *B*, and *F*, respectively, where $\mu_{c(A)}$ is related to μ_A through

$$\mu_{c(A)}(x,y) = \mu_A(x).$$

then

$$\begin{aligned}\mu_{c(A)\cap F}(x,y) &= \min\left[\mu_{c(A)}(x,y), \mu_F(x,y)\right] \\ &= \min\left[\mu_A(x), \mu_F(x,y)\right].\end{aligned}$$

By projecting *c(A)* $\cap$ *F* onto the y-axis, we have

$$\mu_B(y) = \max_x \min\left[\mu_A(x), \mu_F(x,y)\right].$$

This formula reduces to the max-min composition of two relation matrices if both A (a unary fuzzy relation) and F (a binary fuzzy relation) have finite universes of discourse. Conventionally, B is represented as

$$B = A \circ F$$

It is interesting to note that *the extension principle* in Section 3.2.1 is in fact a special case of the compositional rule of inference. Specifically, if $y = f(x)$ is a common crisp one-to-one or many-to-one function, then the derivation of the induced fuzzy set B on Y is exactly what is accomplished by the extension principle.

Using the compositional rule of inference, we can formalize an inference procedure upon a set of fuzzy if-then rules. This inference procedure, generally called approximate reasoning or fuzzy reasoning, is the topic of the next section.

3.4.2 Fuzzy Reasoning

The basic rule of inference in traditional two-valued logic is **modus ponens**, according to which we can infer the truth of a proposition B from the truth of A and the implication $A \rightarrow B$. For instance, if A is identified with "the tomato is red" and B with "the tomato is ripe," then if it is true that "the tomato is red," it is also true that "the tomato is ripe." This concept is illustrated as follows:

Premise 1 (fact):	x is A,
Premise 2 (rule):	if x is A then y is B,
Consequence (conclusion):	y is B.

However, in much of human reasoning, modus ponens is employed in an approximate manner. For example, if we have the same implication rule "if the tomato is red, then it is ripe" and we know that "the tomato is more or less red," then we infer that "the tomato is more or less ripe." This is written as

Premise 1 (fact):	x is A',
Premise 2 (rule):	if x is A then y is B,
Consequence (conclusion):	y is B',

Where A' is close to A and B' is close to B. When A, B, A', and B' are fuzzy sets of appropriate universes, the foregoing inference procedure is called **approximate reasoning** or **fuzzy reasoning**; it is also called generalized modus ponens (**GMP** for short), since it has modus ponens as a special case.

Using the composition rule of inference introduced in the previous section, we can formulate the inference procedure of fuzzy reasoning as the following definition

Definition 3.4 *Approximate reasoning (fuzzy reasoning)*

Let A, A', and B be fuzzy sets of X, X, and Y, respectively. Assume that the fuzzy implication $A \rightarrow B$ is expressed as a fuzzy relation R on $X \times Y$. Then the fuzzy set B induced by "x is A'" and the fuzzy rule "if x is A then y is B" is defined by

$$\begin{aligned}\mu_{B'}(y) &= \max_x \min\left[\mu_{A'}(x), \mu_R(x,y)\right] \\ &= \vee_x \left[\mu_{A'}(x) \wedge \mu_R(x,y)\right],\end{aligned}$$

where $\vee$ and $\wedge$ represent max and min, respectively,

or, equivalently,

$$B' = A' \circ R = A' \circ (A \rightarrow B).$$

Now we can use the inference procedure of fuzzy reasoning to derive conclusions, provided that the fuzzy implication $A \rightarrow B$ is defined as an appropriate binary fuzzy relation.

In what follows, we shall discuss the computational aspects of the fuzzy reasoning introduced in the preceding definition, and then extend the discussion to situations in which multiple fuzzy rules with multiple antecedents are involved in describing a system's behaviour. However, we will restrict our considerations to Mamdani's fuzzy implication functions and the classical max-min composition, because of their wide applicability and easy graphic interpretation.

3.4.2.1 Single Rule with Single Antecedent

This is the simplest case, where

$$\begin{aligned}\mu_{B'}(y) &= \left[\vee_x \left[\mu_{A'}(x) \wedge \mu_A(x)\right]\right] \wedge \mu_B(y) \\ &= w \wedge \mu_B(y).\end{aligned}$$

In other words, first we find the degree of match *w* as the maximum of $\mu_{A'}(x) \wedge \mu_A(x)$ (the shade area in the antecedent part of figure 3.5); then the MF of the resulting B' is equal to the MF of *B* clipped by *w*, Intuitively, *w* represents a measure of degree of belief for the antecedent part of a rule; this measure gets propagated by the if-then rules and the resulting degree of belief or MF for the consequent part (B' in figure 3.5) should be no greater than *w*.

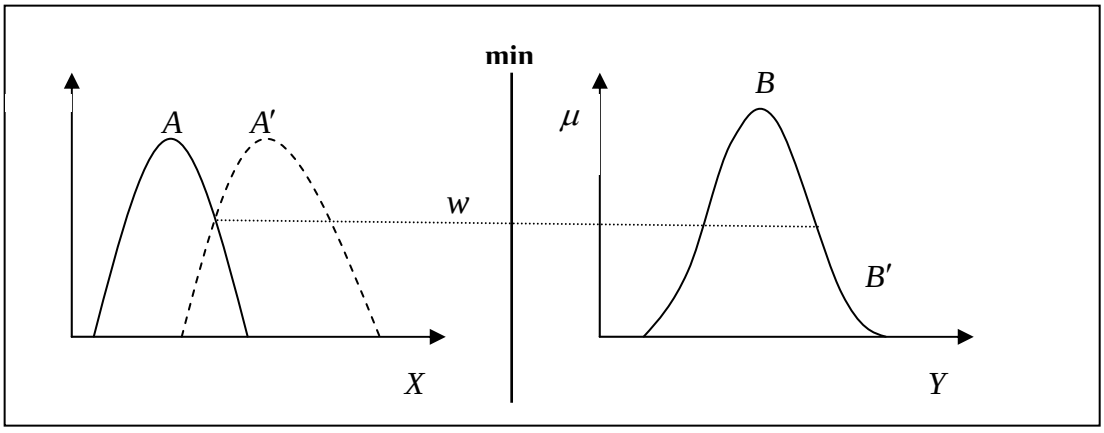


Figure 3.5 Single rules with single antecedent GMP using max-min composition.

3.4.2.2 Single Rule with Multiple Antecedents

A fuzzy if-then rule with two antecedents is usually written as "if *x is A* and *y* is *B* then z is C." The corresponding problem for GMP is expressed as

Premise 1 (fact):	x is A' and y is B' ,
Premise 2 (rule):	if x is A and y is B then z is C,
Consequence (conclusion):	z is C'.

The fuzzy rule in premise 2 can be put into the simpler form "$A \times B \to C$". Intuitively, this fuzzy rule can be transformed into a ternary fuzzy relation R_m base on Mamdani's fuzzy implication function, as follows:

$$R_m(A,B,C) = (A \times B) \times C = \int_{X \times Y \times Z} \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \Big/ (x,y,z).$$

resulting C' is expressed as

$$C' = (A' \times B') \circ (A \times B \to C).$$

Thus

$$\begin{aligned}
\mu_{C'}(z) &= \vee_{x,y}\left[\mu_{A'}(x) \wedge \mu_{B'}(y)\right] \wedge \left[\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)\right] \\
&= \vee_{x,y}\left[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)\right] \wedge \mu_C(z) \\
&= \left\{\vee_x\left[\mu_{A'}(x) \wedge \mu_A(x)\right]\right\} \wedge \left\{\vee_y\left[\mu_{B'}(y) \wedge \mu_B(y)\right]\right\} \wedge \mu_C(z) \\
&= (w_1 \wedge w_2) \wedge \mu_C(z),
\end{aligned}$$

where w_1 and w_2 are the maxima of the MFs of $A \cap A'$ and $B \cap B'$, respectively. In general, w_1 denotes the **degrees of compatibility** between A and A'; similarly for w_2. Since the antecedent part of the fuzzy rule is constructed by the connective "and," $w_1 \wedge w_2$ is called the **firing strength or degree of fulfilment** of the fuzzy

rule**,** which represents the degree to which the antecedent part of the rule is satisfied. A graphic interpretation is shown in figure 3.6, where MF of the resulting C' is equal to the MF of C clipped the firing strength w, $w = w_1 \wedge w_2$.

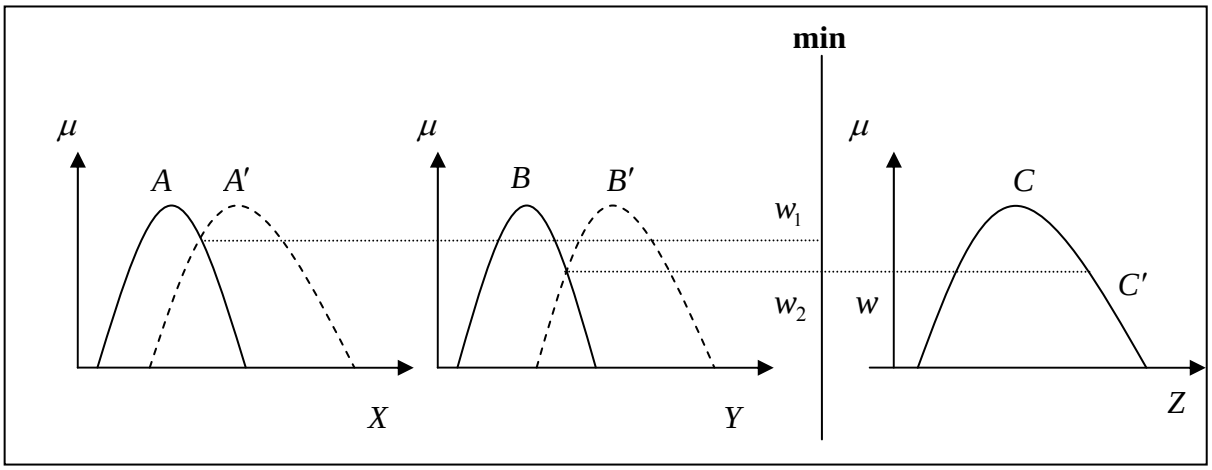


Figure 3.6 Approximate reasoning for multiple antecedents.

3.4.2.3 Multiple Rules with Multiple Antecedents

The interpretation of multiple rules is usually taken as the union of the fuzzy relations corresponding to the fuzzy rules. Therefore, for a GMP problem written as

premise 1 (fact): x is A' and y is B',

premise 2 (rule 1): if x is A_1 and y is B_1 then z is C_1,

premise 3 (rule 2): if x is A_2 and y is B_2 then z is C_2,

---

consequence (conclusion): z is C',

To verify this inference procedure, let $R_1 = A_1 \times B_1 \rightarrow C_1$ and $R_2 = A_2 \times B_2 \rightarrow C_2$. Since the max-min composition operator $\circ$ is distributive over the $\cup$ operator, it follows that

$$
\begin{aligned}
C' &= (A' \times B') \circ (R_1 \cup R_2) \\
&= \left[(A' \times B') \circ R_1\right] \cup \left[(A' \times B') \circ R_2\right] \\
&= C_1' \cup C_2',
\end{aligned}
$$

Where C_1' and C_2' are the inferred fuzzy sets for rules 1 and 2, respectively.

Figure 3.7 shows graphically the operation of fuzzy reasoning for multiple rules with multiple antecedents.

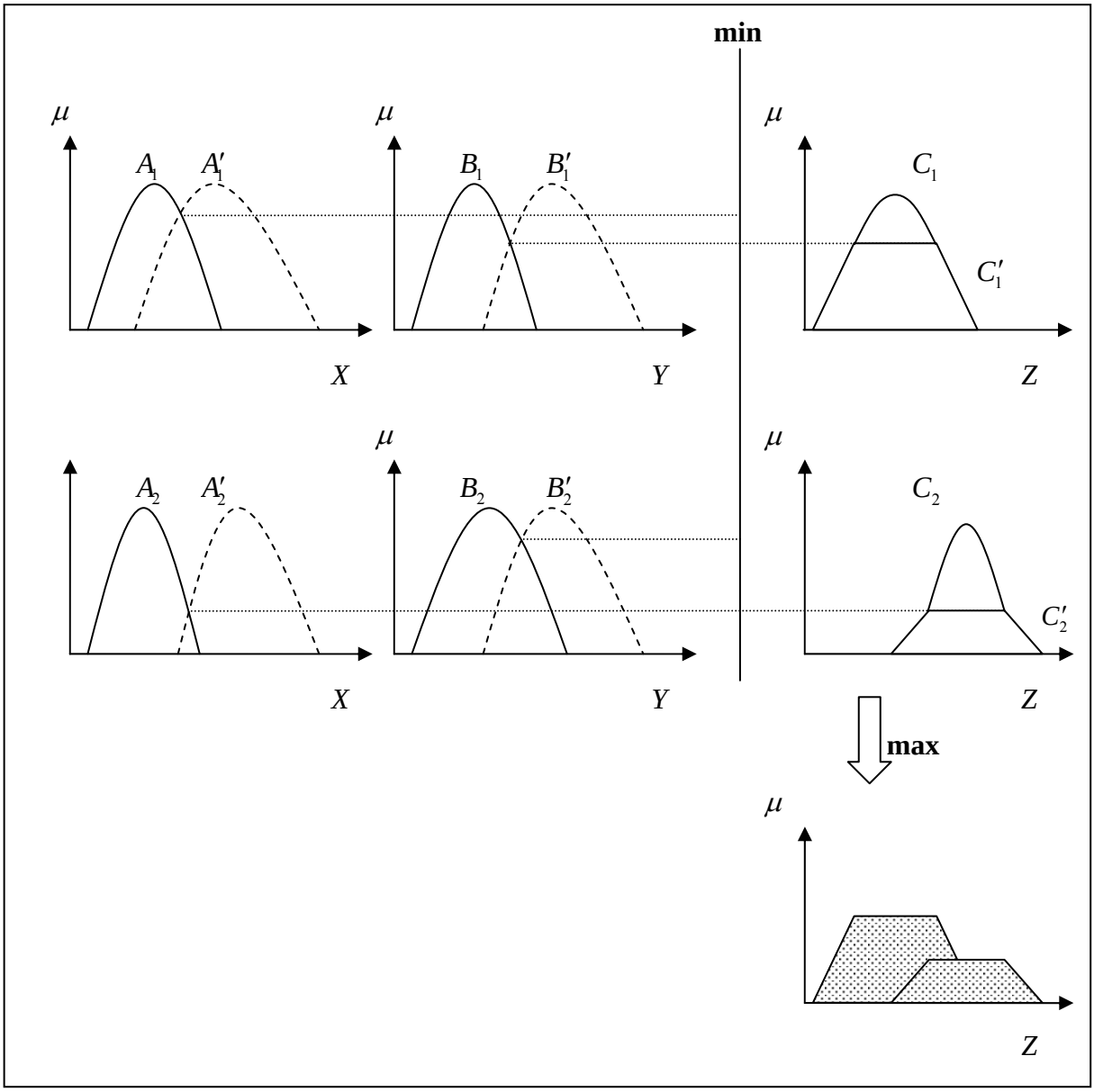


Figure 3.7 Fuzzy reasoning for multiple rules with multiple antecedents

When a given fuzzy rule assumes the form "if *x* is *A* or *y* is *B* then *z* is *C*," then firing strength is given as the maximum of degree of match on the antecedent part for a given condition. This fuzzy rule is equivalent to the union of the two fuzzy rules "if *x* is *A* then *z* is *C*" and "if *y is B* then *z* is *C*."

In summary, the process of fuzzy reasoning or approximate reasoning can be divided into four steps:

1. **Degrees of compatibility** Compare the known facts with the antecedents of fuzzy rules to find the degrees of compatibility with respect to each antecedent MF.
2. **Firing strength** Combine degrees of compatibility with respect to antecedent MFs in a rule using fuzzy AND or OR operators to form a firing strength that indicates the degree to which the antecedent part of the rule is satisfied.
3. **Qualified (induced) consequent MFs** Apply the firing strength to the consequent MF of a rule to generate a qualified consequent MF. (The qualified consequent MFs represent how the firing strength gets propagated and used in a fuzzy implication statement.)
4. **Overall output MF** Aggregate all the qualified consequent MFs to obtain an overall output MF.

These four steps are also employed in a fuzzy inference system, which is introduced in the next chapter.

Chapter 4

4 Fuzzy Inference Systems

4.1 Introduction

The fuzzy inference system is a computer model based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. It has found successful applications in a wide variety of fields, such as automatic control, data classification, decision analysis, expert systems, time series prediction, robotics and pattern recognition. Because of its multidisciplinary nature, the fuzzy inference system is known by various other names, such as **fuzzy-rule-based system** [15], **fuzzy expert system** [15], **fuzzy model** [2], **fuzzy associative memory** [16], **fuzzy logic controller** [11], or simply (and ambiguously) **fuzzy system**.

The basic structure of a fuzzy inference system consists of three conceptual components: a **rule base,** which contains a selection of fuzzy rules; a **database** (or **dictionary**)**,** which defines the membership functions used in the fuzzy rules; and a **reasoning mechanism,** which performs the inference procedure upon the rules and given facts to derive a reasonable output or conclusion (usually the fuzzy reasoning introduced in Section 3.4.2).

Note that the basic fuzzy inference system can take either fuzzy inputs or crisp inputs (which are viewed as fuzzy singletons), but the outputs it produces are almost always fuzzy sets. Sometimes it is necessary to have a crisp output, especially in a situation where a fuzzy inference system is used as a controller. Therefore, we need a method of **defuzzification (**which will be explained in Section 4.2.1) to extract a crisp value that best represents a fuzzy set. A fuzzy inference system with a crisp output is shown in figure 4.1, where the dashed line indicates a basic fuzzy inference system with fuzzy output and the defuzzification block serves the purpose of transforming an output fuzzy set into a crisp single value.

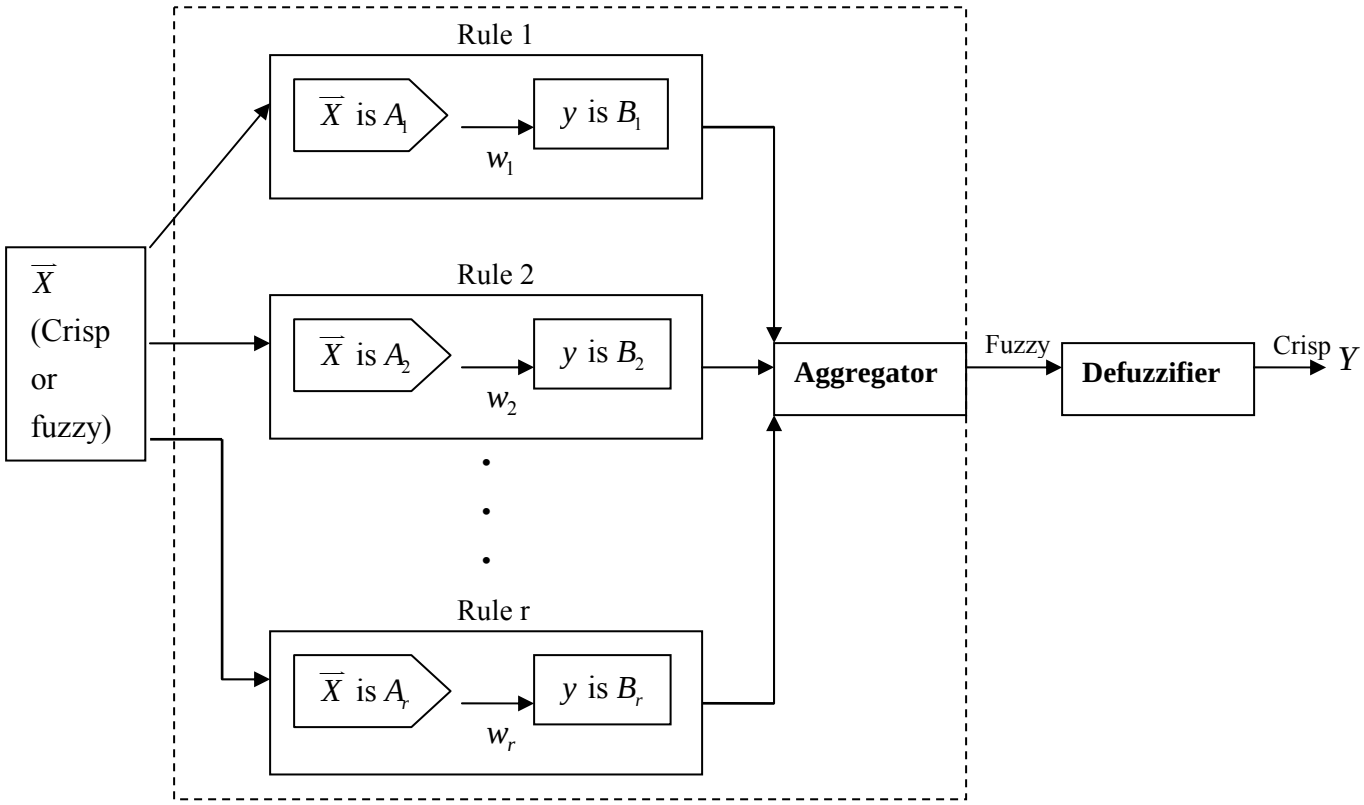


Figure 4.1 Block diagram for a fuzzy inference system

With crisp inputs and outputs, a fuzzy inference system implements a nonlinear mapping from its input space to output space. This mapping is accomplished by a number of fuzzy if-then rules, each of which describes the local behaviour of the mapping. In particular, the antecedent of a rule defines a fuzzy region in the input space, while the consequent specifies the output in the fuzzy region.

In what follows, we shall introduce three types of fuzzy inference systems that have been widely employed in various applications. The differences between these three fuzzy inference systems lie in the consequence of their fuzzy rules, and thus their aggregation and defuzzification procedures differ accordingly.

4.2 Mamdani fuzzy models

The **Mamdani fuzzy inference system** was proposed by E. H. Mamdani [11] as the first attempt to use fuzzy inference system to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators. Figure 4.2 is an illustration of how a two-rule Maindani fuzzy inference system derives the overall output z when subjected to two crisp inputs *x* and *y*.

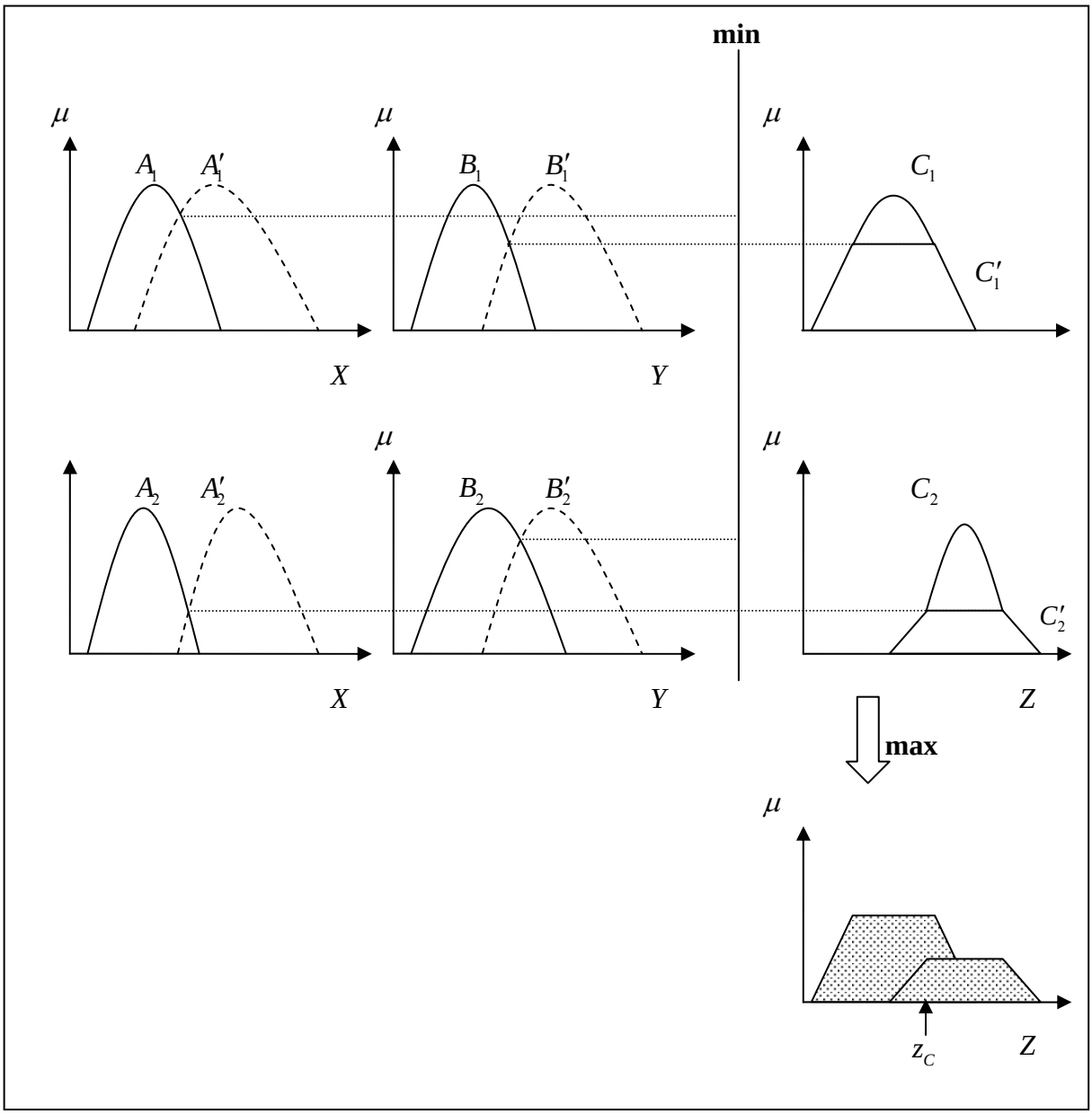


Figure 4.2 The Mamdani fuzzy inference system using min and max for T-norm and T-conorm operators, respectively.

If we adopt max and algebraic product as our choice for the intersection operators T-norm and the union operators T-conorm, respectively, and use max-product composition instead of the original max-min composition, then the resulting fuzzy reasoning is shown in Figure 4.3 (shown in the next section), where the inferred output of each rule is a fuzzy set scaled down by its firing strength via algebraic product. Although this type of fuzzy reasoning was not employed in Mamdani's original paper, it has often been used in the literature. Other variations are possible if we use different T-norm and T-conorm operators.

In Mamdani's application [6], two fuzzy inference systems were used as two controllers to generate the heat input to the boiler and throttle opening of the engine cylinder, respectively, to regulate the steam pressure in the boiler and the speed of the engine. Since the boiler and the engine takes only crisp values as inputs, we have to use a defuzzifier to convert a fuzzy set to a crisp value.

4.2.1 Defuzzification

Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representative value. In general, there are five methods for defuzzifying a fuzzy set A of a universe of discourse Z. A brief explanation of each defuzzification strategy follows.

1. Centroid of area z_c:

$$z_c = \frac{\int_z \mu_A(z)\, z dz}{\int_z \mu_A(z)\, dz},$$

where $\mu_A(z)$ is the aggregated output MF. This is the most widely adopted defuzzification strategy, which is reminiscent of the calculation of expected values of probability distributions.

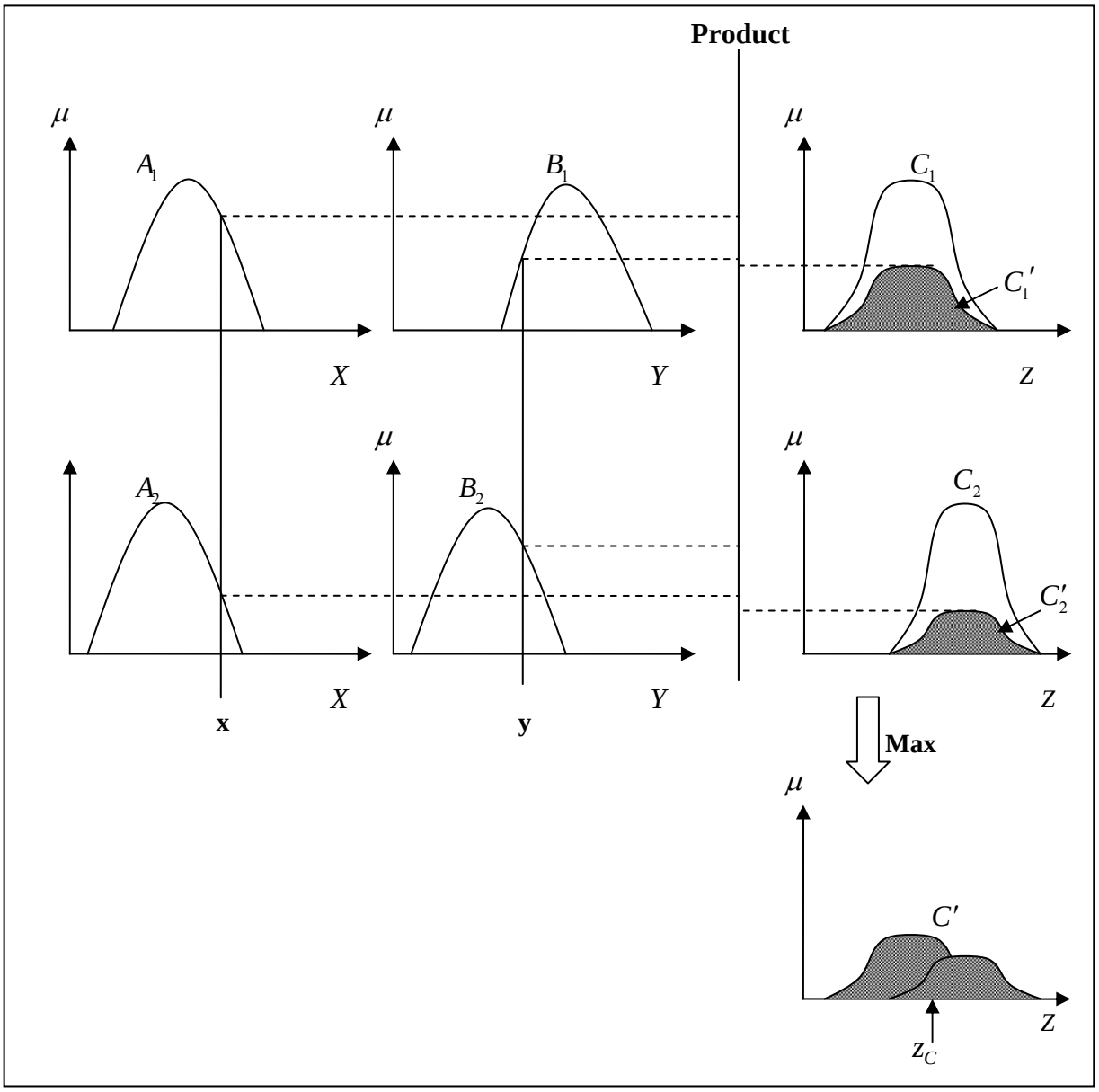


Figure 4.3 The Mamdani fuzzy inference system using product and max for T-norm and T-conorm operators, respectively.

2. Bisector of area, z_B split into two regions with the same area. z_B satisfies

$$\int_{\alpha}^{z_B} \mu_A(z)dz = \int_{z_B}^{\beta} \mu_A(z)dz,$$

where $\alpha = \min\{z | z \in Z\}$ and $\beta = \max\{z | z \in Z\}$, which is the vertical line $z = z_B$ that partitions the region between $z = \alpha$, $z = \beta$, $y = 0$ and $y = \mu_A(z)$.

3. Mean of maximum, z_m is the average of the maximizing z at which the MF reach a maximum μ^*. In symbols,

$$z_m = \frac{\int_{Z'} zdz}{\int_{Z'} dz},$$

where $Z' = \{z \mid \mu_A(z) = \mu^*\}$. In particular, if $\mu_A(z)$ has a single maximum at $z = z^*$, then $z_m = z^*$. The mean of maximum is the defuzzification strategy employed in Mamdani's fuzzy logic controllers [2].

4. Smallest of maximum z_s is the minimum of the magnitude of the maximizing z.

5. Largest of maximum z_l is the maximum of the magnitude of the maximizing z.

Because of their obvious bias, z_s and z_l are not used as often as the other three defuzzification methods.

The calculation needed to carry out any of these five defuzzification operations is time-consuming unless special hardware support is available. Furthermore, these defuzzification operations are not easily subject to rigorous mathematical analysis, so most of the studies are based on experimental results. This leads to the propositions of other types of fuzzy inference systems that do not need defuzzification at all; two of them are introduced in the next section. Other more flexible defuzzification methods can be found in [17, 18, 19].

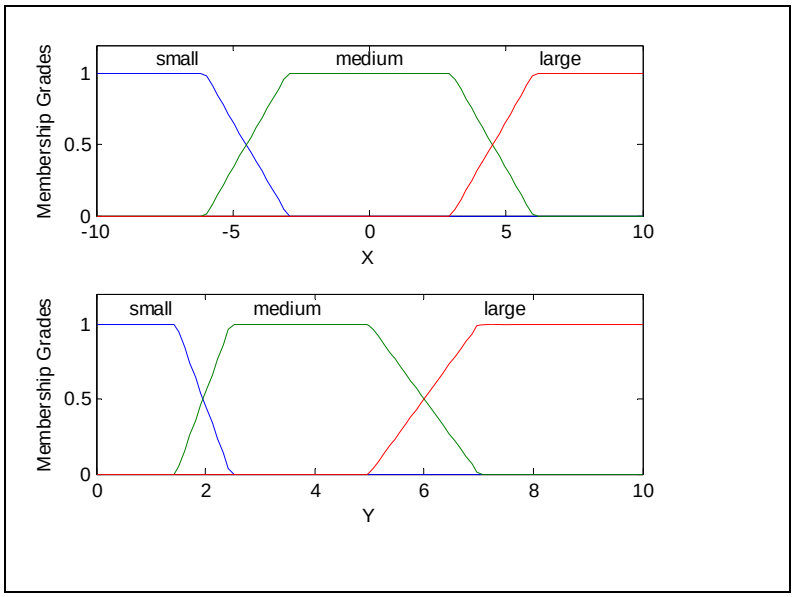
The following two examples are single-input and two-input Mamdani fuzzy models.

Example 4.1 Single-input single-output Mamdani fuzzy model

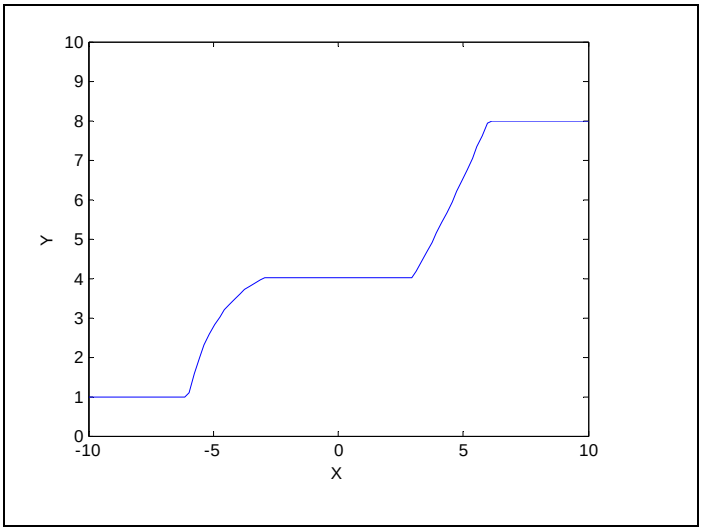
An example of a single-input single-output Mamdani fuzzy model with three rules can be expressed as

- If X is small then Y is small
- If X is medium then Y is medium
- If X is large then Y is large

Figure 4.4(a) plots membership functions of input X and output Y, where the input and output universe are [-10, 10] and [0, 10], respectively. With max-min composition and centroid defuzzification, we can find the overall input-output curve, as shown in figure 4.4(b). Note that the output variable never reaches the maximum (10) and minimum (0) of the output universe. Instead, the reachable minimum and maximum of the output variable are determined by the centroids of the leftmost and rightmost consequent MFs, respectively.



(a)



(b)

Figure 4.5 Single input single output Mamdani fuzzy model in Example 4.1: (a) antecedent and consequent MFs; (b) overall input-output curve

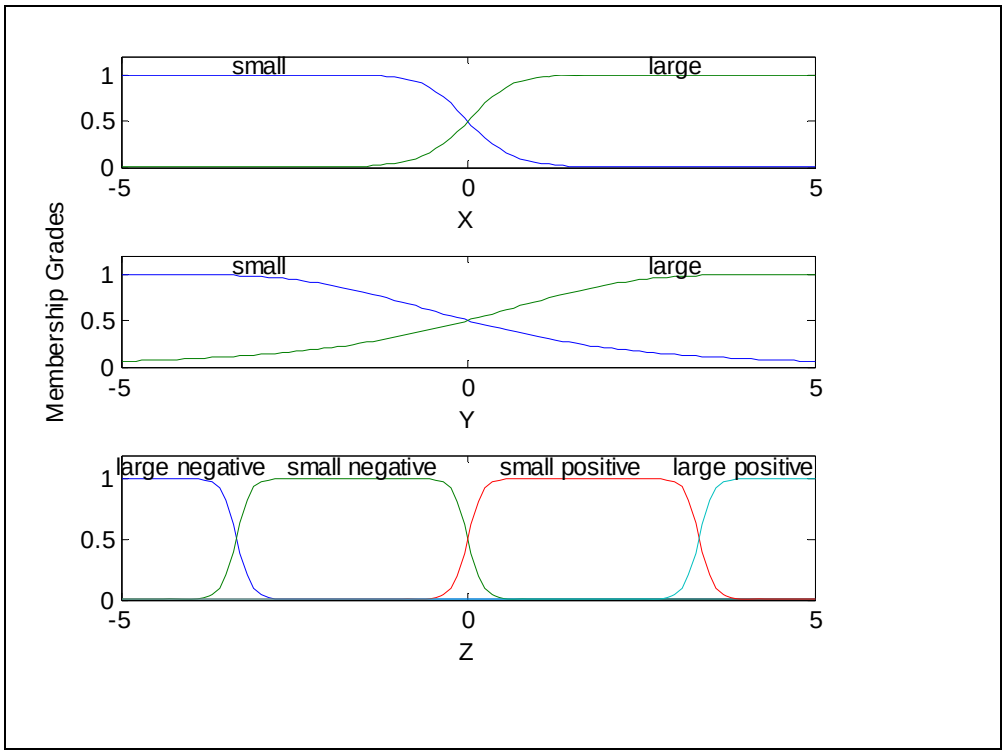
Example 4.2 Two-input single-output Mamdani fuzzy model

An example of a two-input single-output Mamdani fuzzy model with four rules can be expressed as

- If X is small and Y is small then Z is negative large.
- If X is small and Y is large then Z is negative small.
- If X is large and Y is small then Z is positive small.
- If X is large and Y is large then Z is positive large.

Figure 4.6(a) plots membership functions of input X and Y and output Z, all with the same universe [-5, 5]. With max-min composition and centroid defuzzification, we can find the overall input-output surface, as shown in Figure 4.6(b).

(a)



(b)

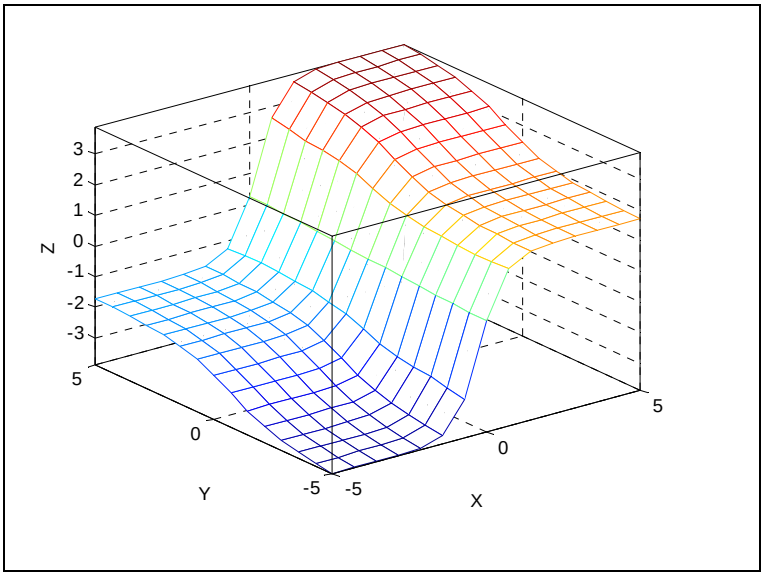


Figure 4.6 Two-input single-output Mamdani fuzzy model in Example 4.2: (a) antecedent and consequent MFs; (b) overall input-output surface.

4.2.2 Other Variants

A fuzzy inference system in practice may have a certain reasoning mechanism that does not follow the strict definition of the compositional rule of inference because of computational efficiency or mathematical tractability.

Therefore, to completely specify the operation of a Mamdani fuzzy inference system, we need to assign a function for each of the following operators:

- **AND operator** (usually T-norm) for calculating the firing strength of a rule with AND'ed antecedents.

- **OR operator** (usually T-conorm) for calculating the firing strength of a rule with OR'ed antecedents.

- **Implication operator** (usually T-norm) for calculating qualified consequent MFs based on given firing strength.

- **Aggregate operator** (usually T-conorm) for aggregating qualified consequent MFs to generate an overall output MF.

- **Defuzzification operator** for transforming an output MF to a crisp single output value.

One such example is to use product for the implication operator and point-wise summation (sum) for the aggregate operator. (Note that sum is not even a T-conorm operator.) An advantage of this **sum-product composition** [3] is that the final crisp output via centroid defuzzification is equal to the weighted average of the centroids of consequent MFs, where the weighting factor for each rule is equal to its firing strength

multiplied by the area of the consequent MF. This property is expressed as the following theorem.

Theorem 4.1 *Computation shortcut for Mamdani fuzzy inference systems*

Under sum-product composition, the output of a Mamdani fuzzy inference system with centroid defuzzification is equal to the weighted average of the centroids of consequent MFs, where each of the weighting factors is equal to the product of a firing strength and the consequent MF's area.

□

The following example will explain the mechanism of Theorem 4.1.
Let assume we have a fuzzy inference system with two rules. By using product and sum for implication and aggregate operators, respectively, we have

$$\mu_{c'}(z) = w_1 \mu c_1(z) + w_2 \mu c_2(z).$$

(Note that the preceding MF could have values greater than 1 at certain points.) The crisp output under centroid defuzzification is

$$\begin{aligned} z_c &= \frac{\int_Z \mu c'(z) z dz}{\int_Z \mu c'(z) dz} \\ &= \frac{w_1 \int \mu c_1(z) z dz + w_2 \int \mu c_2(z) z dz}{w_1 \int \mu c_1(z) dz + w_2 \int \mu c_2(z) dz} \\ &= \frac{w_1 a_1 z_1 + w_2 a_2 z_2}{w_1 a_1 + w_2 a_2}, \end{aligned}$$

where $a_i = \int_z \mu c_i(z)\,dz$ and $z_i = \frac{\int_Z \mu c_i(z) z\,dz}{\int_Z \mu c_i(z)\,dz}$ are the area and centroid of the consequent MF $\mu c_i(z)$, respectively.

By using this theorem, computation is more efficient if we can obtain the area and centroid of each consequent MF in advance.

4.3 Sugeno Fuzzy Models

The **Sugeno fuzzy model** (also known as the **TSK fuzzy model**) was proposed by Takagi, Sugeno [2] in an effort to develop a systematic approach to generating fuzzy rules from a given input-output data set. A typical fuzzy rule in a Sugeno fuzzy model has the form

$$\text{If } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = f(x, y),$$

Where *A* and *B* are fuzzy sets in the antecedent, while $z = f(x, y)$ is a crisp function in the consequence. Usually $z = f(x, y)$ is a polynomial in the input variables *x* and *y*, but it can be any function as long as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule. When $f(x, y)$ is a first-order polynomial, the resulting fuzzy inference system is called **a first-order Sugeno fuzzy model** which was originally proposed in [2]. For multiple fuzzy rules $i = 1 \ldots n$

$$\text{if } x \text{ is } A_i \text{ and } y \text{ is } B_i \text{ then } f_i$$

the model can be written as

$$z(x, y) = \sum_{i=1}^{n} A_i(x) B_i(y)\, f_i(x, y)$$

When f is a constant, we then have a **zero-order Sugeno fuzzy model,** which can be viewed either as a special case of the Mamdani fuzzy inference system, in which each rule's consequent is specified by a fuzzy singleton (or a pre-defuzzified consequent), or a special case of the Tsukamoto fuzzy model (to be introduced in section 4.4), in which each rule's consequent is specified by an MF of a step function centered at a constant. Moreover, a zero-order Sugeno fuzzy model is functionally equivalent to a radial basis function network under certain minor constraints [20].

The output of a zero-order Sugeno model is a smooth function of its input variables as long as the neighboring MFs in the antecedent have enough overlap. In other words, the overlap of MFs in the consequent of a Mamdani model does not have a decisive effect on the smoothness; it is the overlap of the antecedent MFs that determines the smoothness of the resulting input-output behavior.

Figure 4.7 shows the fuzzy reasoning procedure for a first-order Sugeno fuzzy model. Since each rule has a crisp output, the overall output is obtained via **weighted average,** thus avoiding the time-consuming process of defuzzification required in a Mamdani model. In practice; the weighted average operator is sometimes replaced with the **weighted sum** operator (that is, $z = w_1 z_1 + w_2 z_2$ in Figure 4.7) to reduce computation further, especially in the training of a fuzzy inference system. However, this simplification could lead to the loss of MF linguistic meanings unless the sum of firing strengths (that is, $\sum_i w_i$) is close to unity.

Since the only fuzzy part of a Sugeno model is in its antecedent, it is easy to demonstrate the distinction between a set of fuzzy rules and non-fuzzy ones.

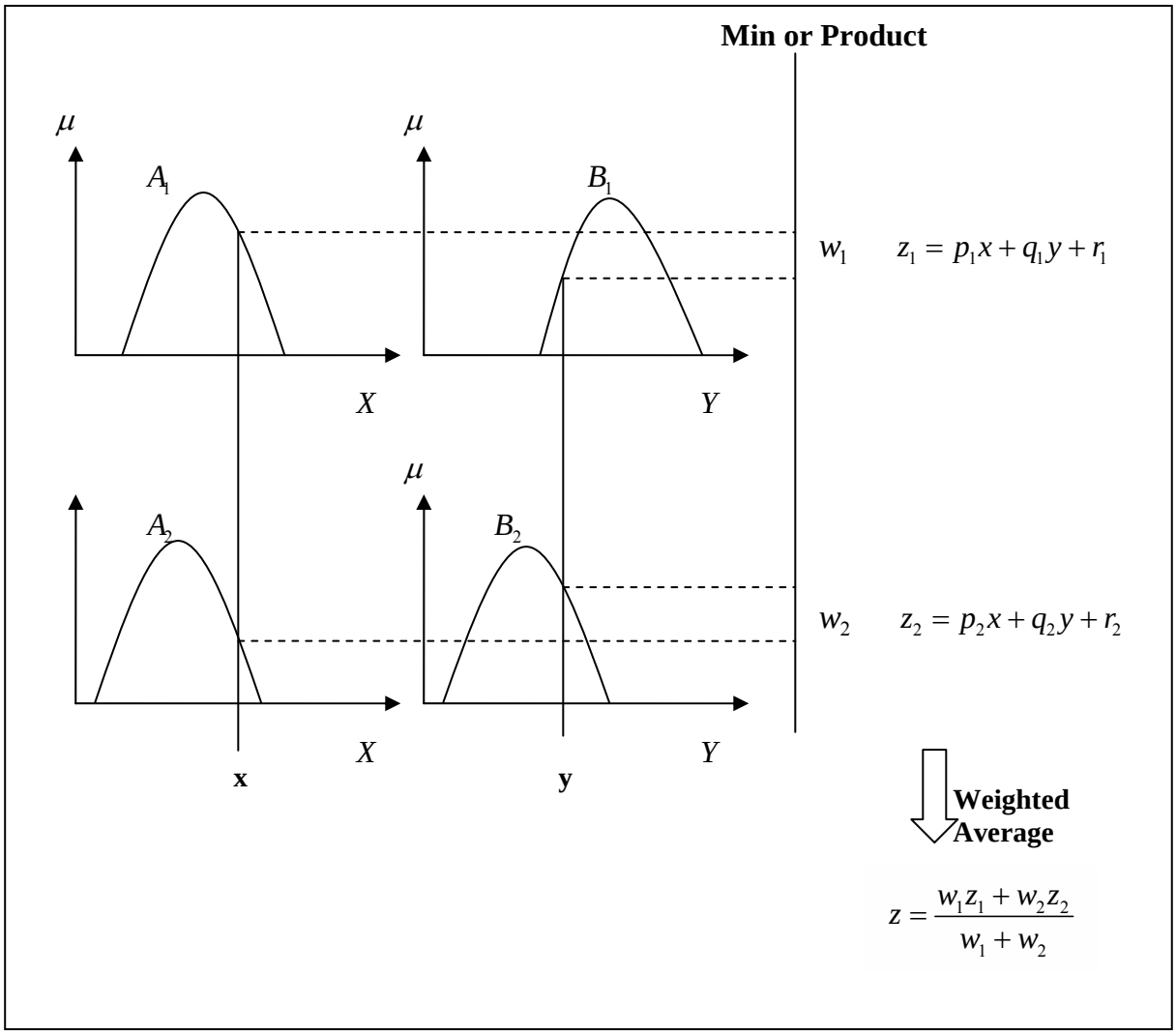


Figure 4.7 The Sugeno fuzzy model.

Example 4.3 A comparison between fuzzy and non-fuzzy rule set

An example of a single-input Sugeno fuzzy model can be expressed as

- If X is small then Y = 0.1X + 6.4.
- If X is medium then Y = -0.5X + 4.
- If X is large then Y = X - 2.

If "small," "medium," and "large" are non-fuzzy sets with membership functions shown in Figure 4.8(a), then the overall input-output curve is piecewise linear, as shown in Figure 4.8(b). On the other hand, if we have smooth membership functions [Figure 4.8(c)] instead, the overall input-output curve [Figure 4.8(d)] becomes a smoother one.

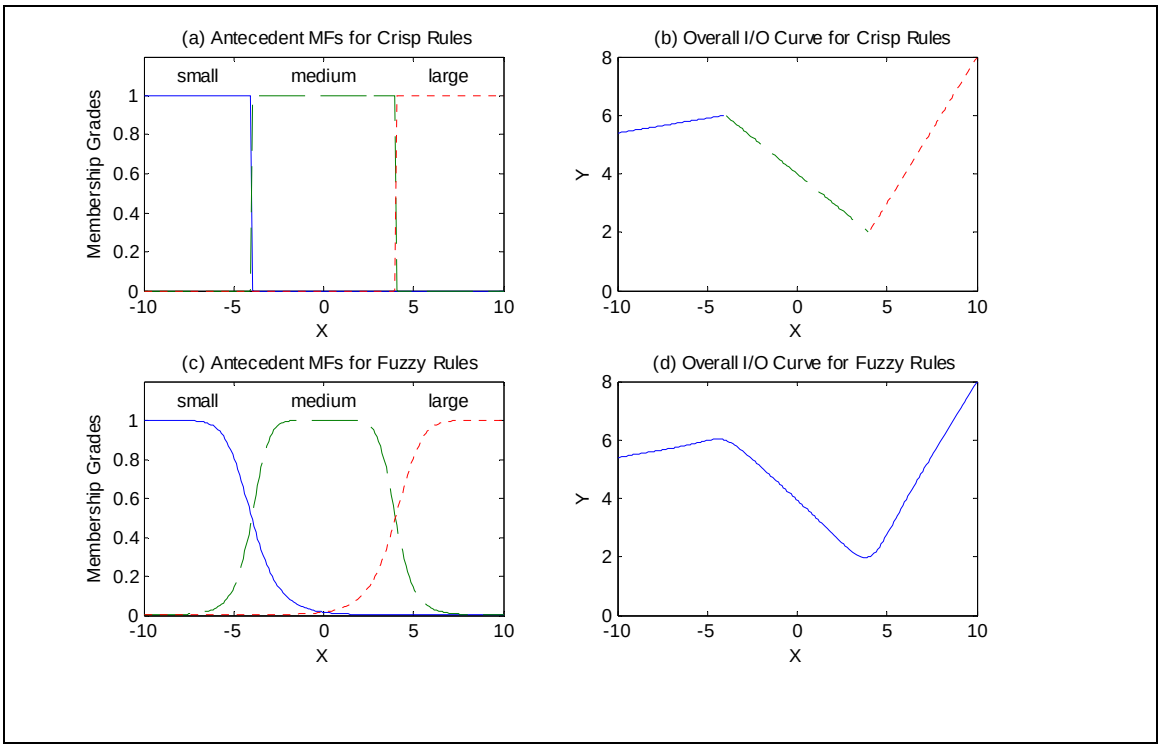


Figure 4.8 Comparison between fuzzy and non-fuzzy rules in example 4.3: (a) Antecedent MFs and (b) input-output curve for non-fuzzy rules; (c) Antecedents MFs and (d) input-output curve for fuzzy rules

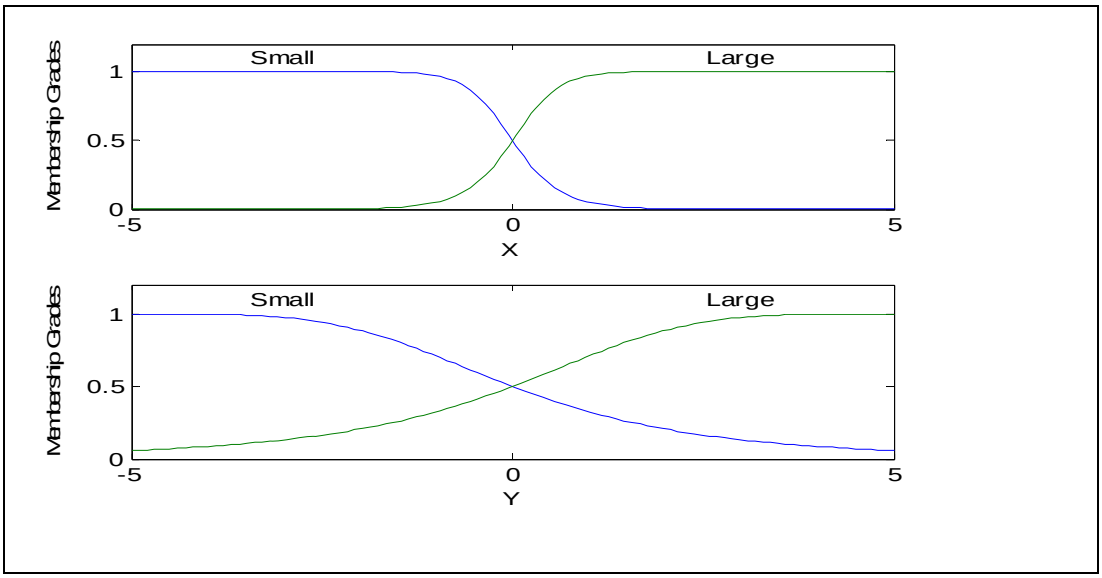
□

Example 4.4 Two-input single-output Sugeno fuzzy model

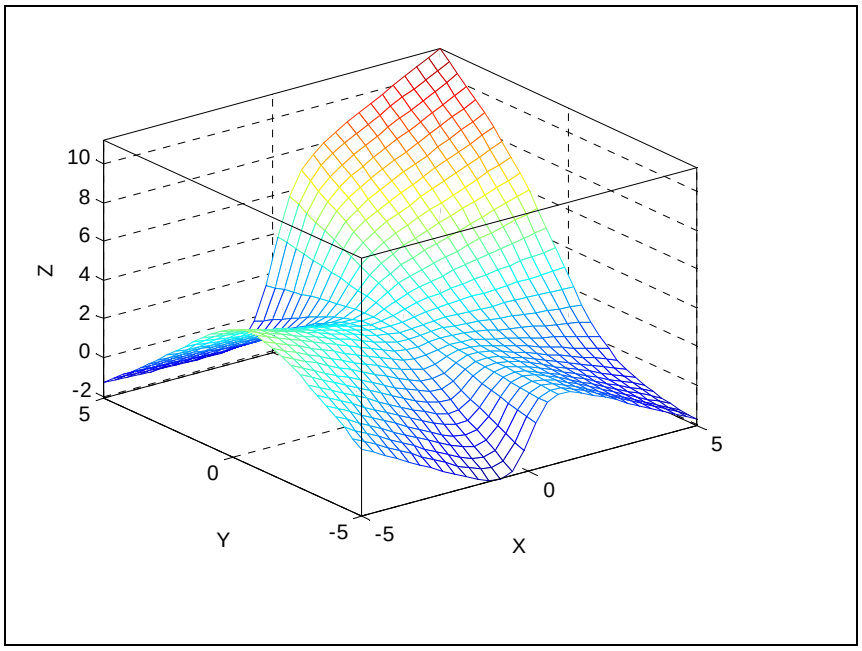
An example of a two-input single-output Sugeno fuzzy model with four rules can be expressed as

- If X is small and Y is small then Z = -X+Y-1.
- If X is small and Y is large then Z = -Y+3.
- If X is large and Y is small then Z = -X+3.
- If X is large and Y is large then Z = X+Y+2.

Figure 4.9(a) plots the membership functions of input X and Y, and Figure 4.9(b) is the resulting input-output surface. The surface is complex, but it is can be seen that the surface is composed of four planes, each of which is specified by the output equation of a fuzzy rule.



(a)



(b)

Figure 4.9 Two-input single-output Sugeno fuzzy model in Example 4.4: (a) antecedent and consequent MFs; (b) overall input-output surface.

□

4.4 Tsukamoto Fuzzy Models

In the **Tsukamoto fuzzy models** [13]**,** the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonic MF, as shown in Figure 4.10. As a result, the inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's output. Figure 4.10 illustrates the reasoning procedure for a two-input two-rule system.

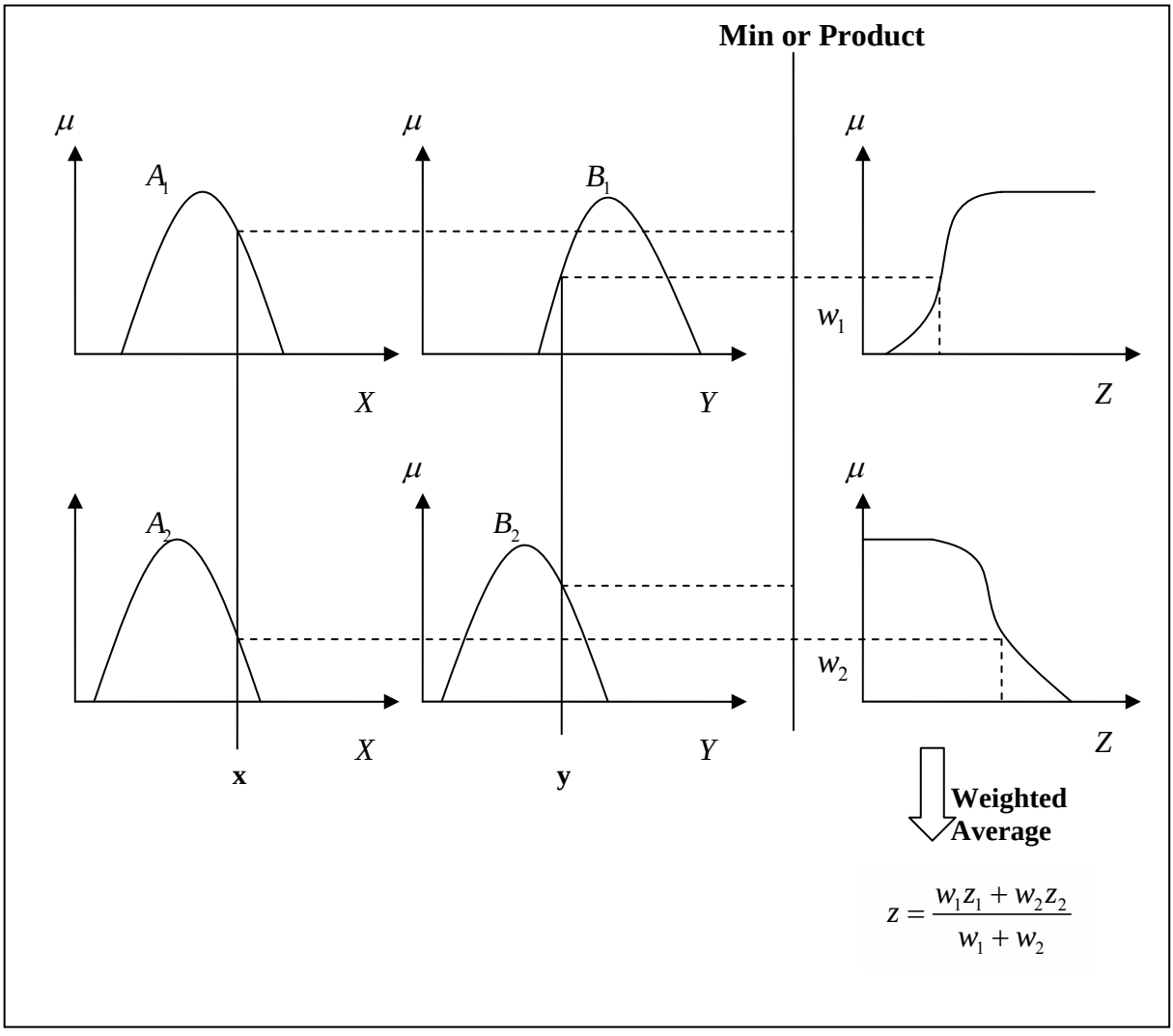


Figure 4.10 The Tsukamoto fuzzy model.

Since each rule infers a crisp output, the Tsukamoto fuzzy model aggregates each rule's output by the method of weighted average and thus avoids the time-consuming process of defuzzification.

An example of a single-input Tsukamoto fuzzy model can be expressed as

- If X is small then Y is C1
- If X is medium then Y is C2
- If X is large then Y is C3,

where the antecedent MFs for “small,” “medium,” and "large" are shown in Figure 4.11(a); and the consequent MFs for “*C1*,” “*C2*,” and “*C3*” are shown in Figure 4.11(b). The overall input-output curve, as shown in Figure 4(d), is equal to $\frac{\sum_{i=1}^{3} w_i f_i}{\sum_{i=1}^{3} w_i}$, where f_i is the output of each rule induced by the firing strength w_i and MF for *Ci*. If we plot each rule's output f_i as a function of *x*, we obtain Figure 4.11(c), which is not quite obvious from the original rule base and MF plots.

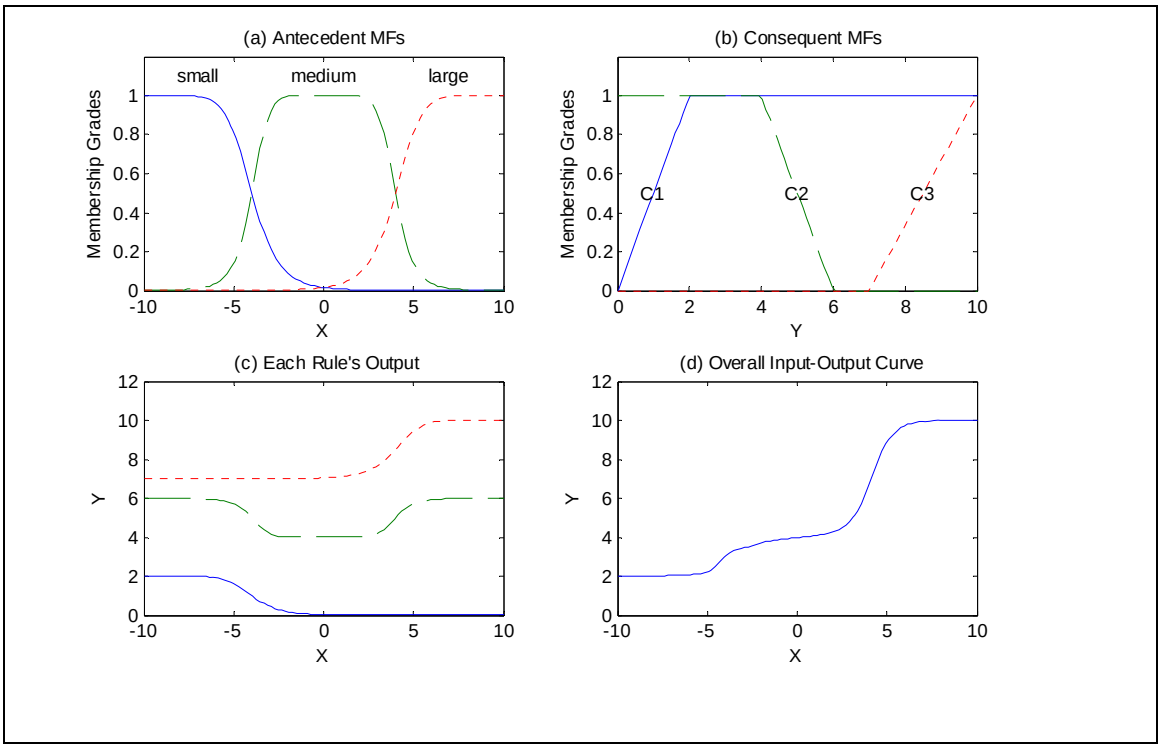


Figure 4.11 Single input single output Tsukamoto fuzzy model

Since the reasoning mechanism of the Tsukamoto fuzzy model does not follow strictly the compositional rule of inference, the output is always crisp even when the inputs are fuzzy.

4.5 Summary

This chapter presents three frequently used fuzzy inference systems: the Mamdani, Sugeno, and Tsukamoto fuzzy models. These systems are the most important modeling tool based on fuzzy set theory. Conventional fuzzy inference systems are typically built by domain experts and have been used in automatic control, decision analysis, and expert systems.

Optimization and adaptive technique expand the application of fuzzy inference systems to fields such as adaptive control, adaptive signal processing, nonlinear regression, and pattern recognition. In Chapter five we shall discuss such techniques, especially the back propagation learning rule which is used in adaptive based fuzzy inference systems.

Chapter 5

5 Adaptive Networks

5.1 Introduction

This chapter describes the architectures and learning procedures of adaptive networks, a unifying framework that is used in almost all kinds of neural network paradigms with supervised learning capabilities. The fundamentals of adaptive networks will be a key element in understanding other neural network paradigms.

An adaptive network is a network structure consisting of a number of nodes connected through directional links. Each node represents a process unit, and the links between nodes specify the causal relationship between the connected nodes. All or parts of the nodes are adaptive, which means the outputs of these nodes depend on modifiable parameters pertaining to these nodes. The learning rule specifies how these parameters should be updated to minimize a prescribed error measure, which is a mathematical expression that measures the discrepancy between the network's actual output and a desired output.

The basic learning rule of the adaptive network is the well-known steepest descent method, in which the gradient vector is derived by successive invocations of the chain rule. This method for systematic calculation of the gradient vector was proposed independently, by Bryson and Ho [21] in 1969, Werbos [4] in 1974, and Parker [22] in 1982. However, because research on artificial neural networks was still in its infancy at those times, their early work failed to receive the attention it deserved. In 1986, Rumelhart et al. [3] used the same procedure to find the gradient in a multilayer neural network. Their procedure was called the back-propagation learning rule, a name which is now widely known because the work of Rumelhart et al. inspired enormous interest in research on neural networks. In this chapter, we introduce Werbos's original back-propagation method for finding gradient vectors and also present an improved

version by Jang [23, 1] which speed up the time-consuming learning process by incorporating the least-squares method.

5.2 Architecture

As the name implies an **adaptive network** (Figure 5.1) is a network structure whose overall input-output behaviour is determined by a collection of modifiable parameters. Specifically, the configuration of an adaptive network is composed of a set of nodes connected by directed links, where each node performs a static **node function** on its incoming signals to generate a single **node output** and each link specifies the direction of signal flow from one node to another. Usually a node function is a parameterized function with modifiable parameters: by changing these parameters, we change the node function as well as the overall behaviour of the adaptive network.

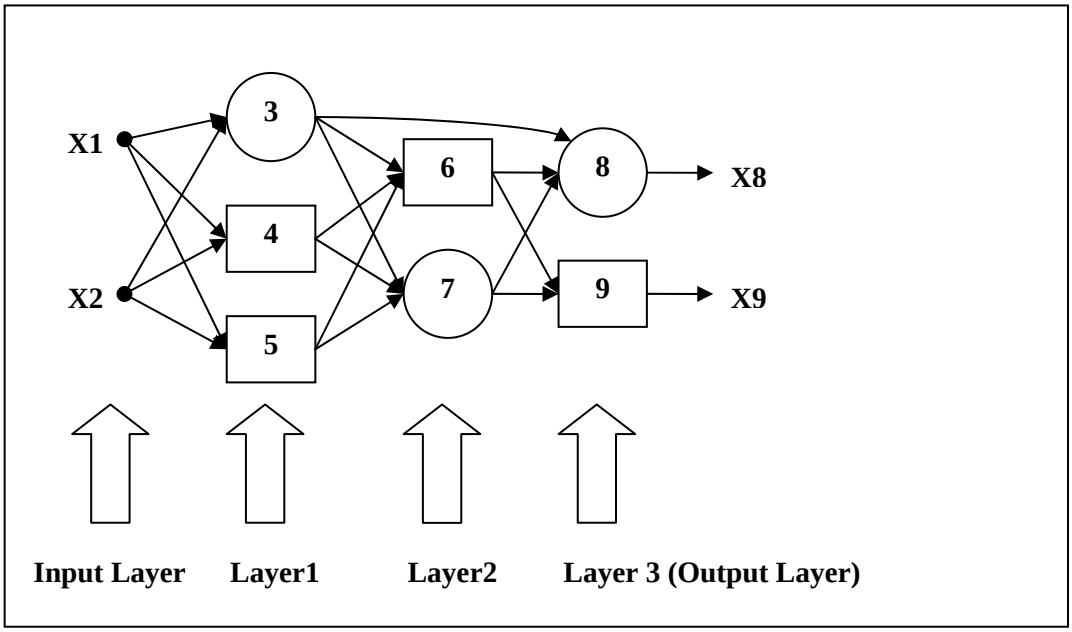


Figure 5.1 A feed-forward adaptive network in layered representation

In the following discussion, we shall assume that each node in an adaptive network performs a static mapping from its input(s) to output. Namely, a node's output depends on its current inputs only: there are no dynamics or internal states in each

node. Moreover, to facilitate the understanding of learning algorithms, we assume that all node functions are differentiable except at a finite number of points. In the most general case, an adaptive network is heterogeneous that is each node may have a specific node function different from the others. Links in an adaptive network are merely used to specify the propagation direction of node outputs: generally there are no weights or parameters associated with links. Figure 5.1 is a typical adaptive network with two inputs and two outputs.

The parameters of an adaptive network are distributed into its nodes, so each node has a local parameter set. The union of these local parameter sets is the network's overall parameter set. If a node's parameter set is not empty, then its node function depends on the parameter values; we use a square to represent this kind of **adaptive node.** On the other hand, if a node has an empty parameter set, then it has a fixed function; we use a circle to denote this type of **fixed node.** Each adaptive node can be decomposed into a fixed node plus one or several parameter nodes, as illustrated in the following example.

Example 5.1 Parameter sharing in adaptive networks

Figure 5.2(a) shows an adaptive network with only one node, which can be represented as $y = f(x, a)$, where x and y are the input and output, respectively, and a is the parameter of the node. An equivalent representation is to move the parameter out of the node and put it into a **parameter node**, as shown in Figure 5.2(a). It is obvious that a parameter node is a special case of an adaptive node in which there are no inputs and the output is the parameter itself. The parameter node is useful in solving certain representation problems, such as the **parameter sharing problem** in Figure 5.2(b), where two adaptive nodes $u = g(x, a)$ and $v = h(y, a)$ share the same parameter a, as denoted by the dotted line linking these two nodes. By taking out the parameter and putting it into a parameter node, we can embed the parameter sharing requirement into the architecture. This simplifies network representation as well as software implementation.

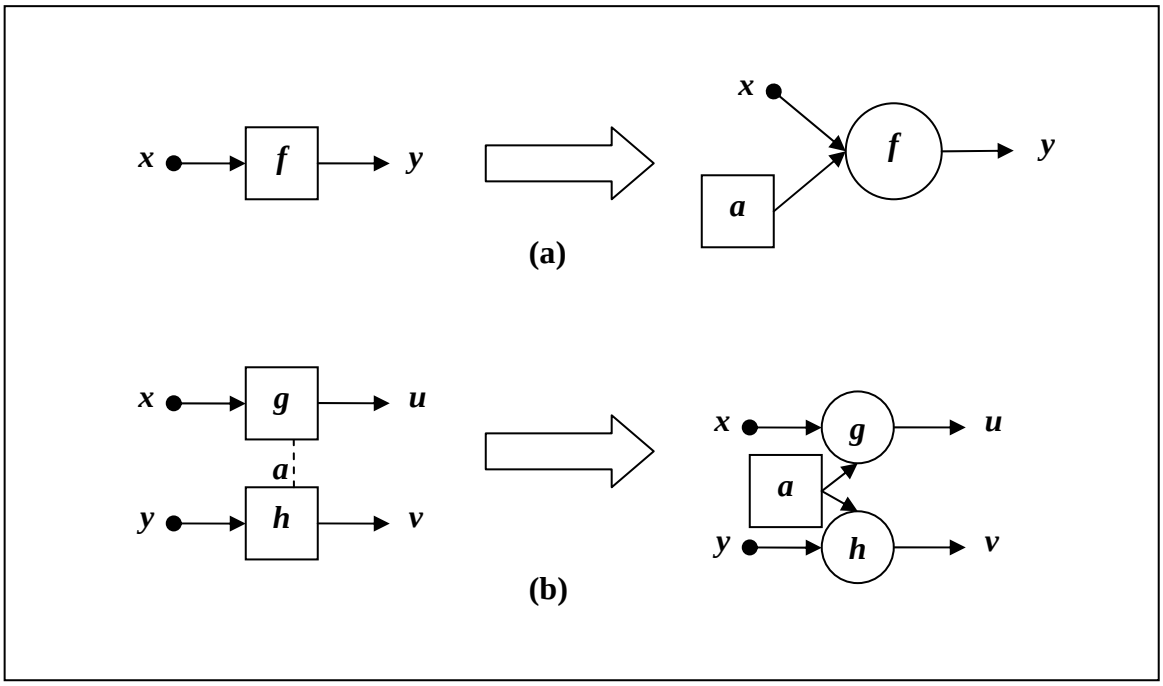


Figure 5.2 Decomposition of adaptive nodes: (a) a single node; (b) parameter sharing problem.

Adaptive networks are generally classified into two categories on the basis of the type of connections they have: **feed-forward** and **recurrent.** The adaptive network shown in Figure 5.1 is feed-forward, since the output of each node propagates from the input side (left) to the output side (right) unanimously. If there is a feedback link that forms a circular path in a network, then the network is recurrent; Figure 5.3 is an example. (From the viewpoint of graph theory, a feed-forward network is represented by an *acyclic* directed graph which contains no directed cycles, while a recurrent network always contains at least one directed cycle.)

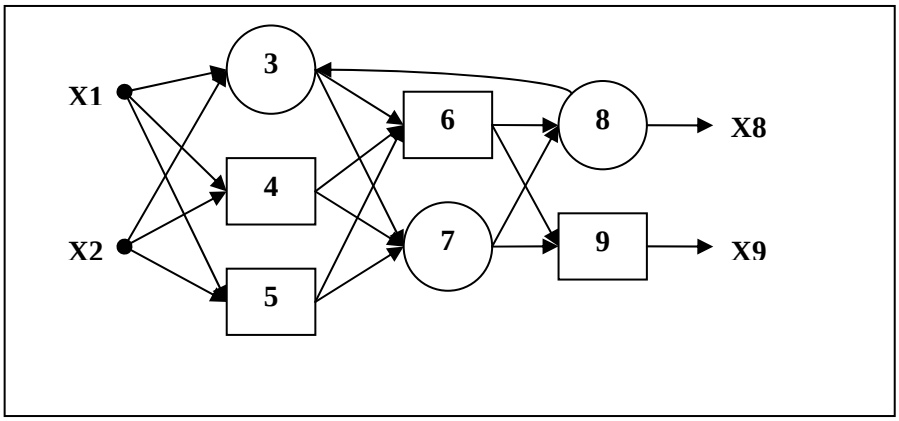


Figure 5.3 A recurrent adaptive network.

In the **layered representation** of the feed-forward adaptive network in Figure 5.1, there are no links between nodes in the same layer, and outputs of nodes in a specific layer are always connected to nodes in succeeding layers. This representation is usually preferred because of its modularity, in that nodes in the same layer have the same functionality or generate the same level of abstraction about input vectors. Another representation of feed-forward networks is the **topological ordering representation,** which labels the nodes in an ordered sequence 1, 2, 3, ..., such that there are no links from node i to node j whenever $i > j$. Figure 5.4 is the topological ordering representation of the network in Figure 5.1. This representation is less modular than the layer representation, but it facilitates the formulation of learning rules, as will be detailed in the next section. (Note that the topological ordering representation is in fact a special case of the layered representation. with one node per layer.)

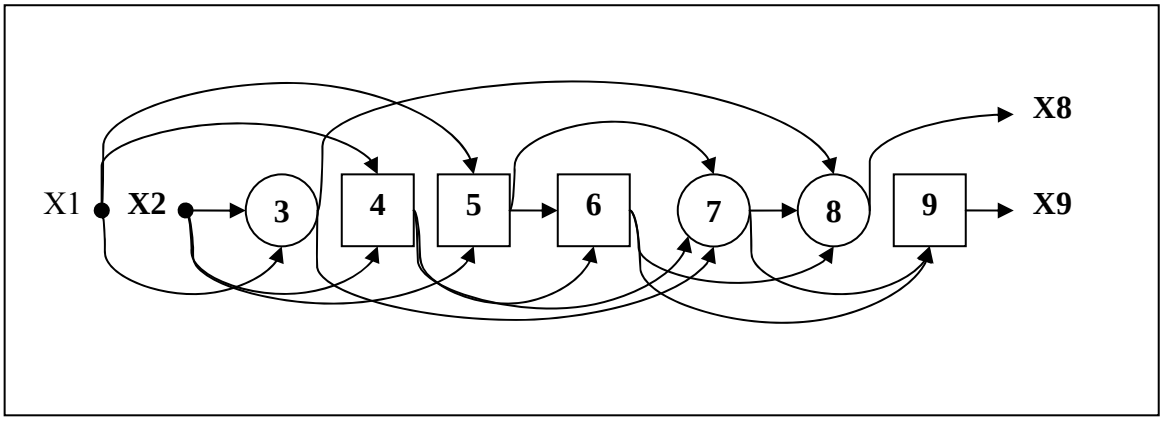


Figure 5.4 A feed-forward adaptive network in topological ordering representation.

Conceptually, a feed-forward adaptive network is actually a static mapping between its input and output spaces: this mapping may be either a simple linear relationship or a highly nonlinear one, depending on the network structure (node arrangement and connections, and so on) and the functionality for each node. Here our aim is to construct a network for achieving a desired nonlinear mapping that is regulated by a data set consisting of desired input-output pairs of a target system to be modelled. This data set is usually called the **training data set,** and the procedures we follow in adjusting the parameters to improve the network's performance are often referred to as the **learning rules** or **adaptation algorithms.** Usually a network's performance is measured as the discrepancy between the desired output and the network's output under the same input conditions. This discrepancy is called the **error measure** and it can assume different forms for different applications. Generally speaking, a learning rule is derived by applying a specific optimization technique to a given error measure.

Before introducing a basic learning algorithm for adaptive networks, we shall present several examples of adaptive networks.

Example 5.2 An adaptive network with a single linear node

Figure 5.5 is an adaptive network with a single node specified by

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3,$$

where x_1 and x_2 are inputs and $a_1, a_2,$ and a_3 are modifiable parameters. The function defines a plane in $x_1 - x_2 - x_3$ space, and by setting appropriate values for the parameters, we can place this plane arbitrarily. By adopting the squared error as the error measure for this network, we can identify the optimal parameters via the linear least-squares estimation method.

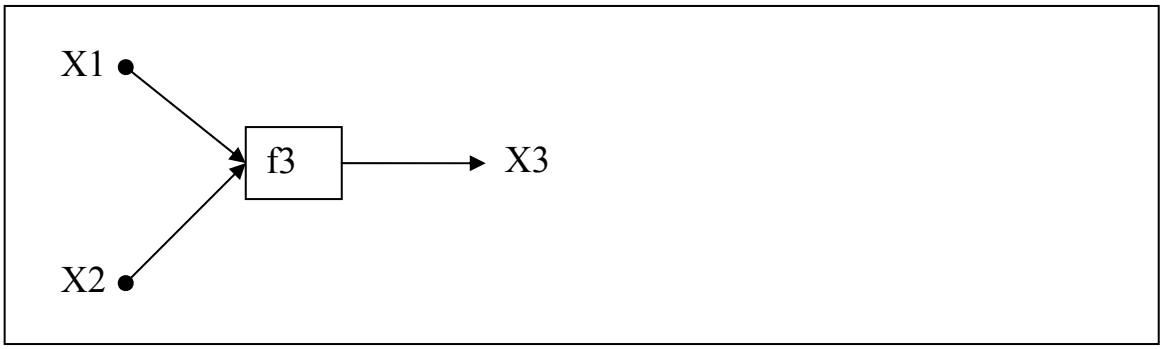


Figure 5.5 A linear single-node adaptive network.

Example 5.3 Perceptron network

If we add another node to let the output of the adaptive network in Figure 5.5 has only two values 0 and 1; then the nonlinear network shown in Figure 5.6 is obtained. Specifically, the node outputs are expressed as

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3,$$

and

$$x_4 = f_4(x_3) = \begin{cases} 1 & \text{if } x_3 \geq 0 \\ 0 & \text{if } x_3 < 0 \end{cases},$$

where f_3 is a linearly parameterized function and f_4 is a step function which maps x_3 to either 0 or 1.

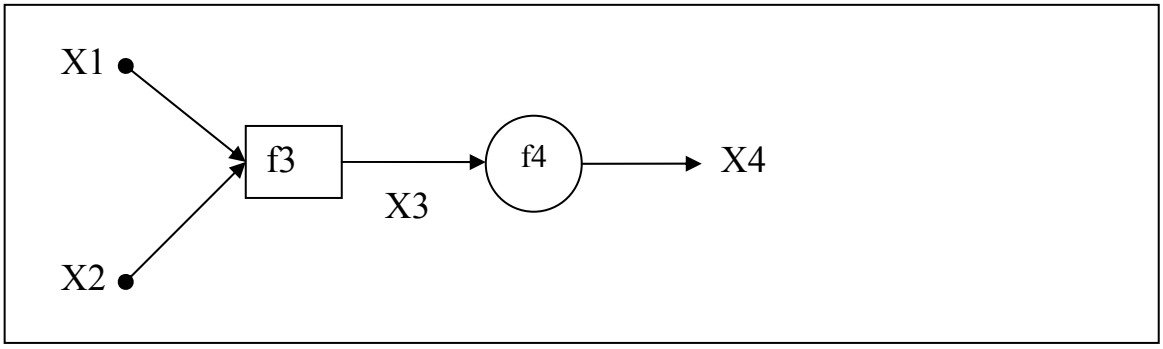


Figure 5.6 A nonlinear single-node adaptive network.

The overall function of this network can be viewed as a **linear classifier** by making a classification decision based on the value of the linear combination of the features. The first node forms a decision boundary as a straight line in $x_1 - x_2$ space; and the second node indicates which half plane the input vector (x_1, x_2) resides in. Obviously, we can form an equivalent network with a single node whose function is the composition of f_3 and f_4; the resulting node is the building block of the classical **perceptron** [24, 25] which is a type of adaptive network invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt. It can be seen as the simplest kind of feed-forward adaptive network: a linear classifier.

Since the step function is discontinuous at one point and flat at all the other points, it is not suitable for derivative-based learning procedures. One way to get around this difficulty is to use the **sigmoidal function** as a squashing function that has values between 0 and 1:

$$x_4 = f_4(x_3) = \frac{1}{1+e^{-x_3}}.$$

This is a continuous and differentiable approximation to the step function. The composition of f_3 and this differentiable f_4 is the building block for the multilayer perceptron in the following example.

Example 5.4 A multilayer perceptron

Figure 5.7 is a typical architecture for a multilayer perceptron with three inputs, two outputs, and three hidden nodes that do not connect directly to either inputs or outputs. Each node in a network of this kind has the same node function, which is the composition of a linear f_3 and a sigmoidal f_4 in Example 5.3. For instance, the node function of node 7 in Figure 5.7 is

$$x_7 = \frac{1}{1+\exp\left[-\left(w_{4,7}x_4 + w_{5,7}x_5 + w_{6,7}x_6 + t_7\right)\right]},$$

where x_4, x_5 and x_6 are outputs from nodes 4, 5, and 6, respectively, and the parameter set of node 7 is denoted by $\{w_{4,7}, w_{5,7}, w_{6,7}, t_7\}$. Usually we view $w_{i,j}$ as the **weight** associated with the link connecting node *i* and *j* and t_j as the **threshold** associated with node *j*. However, this weight-link association is only valid in this type of network. In general, a link only indicates the signal flow direction between connected nodes.

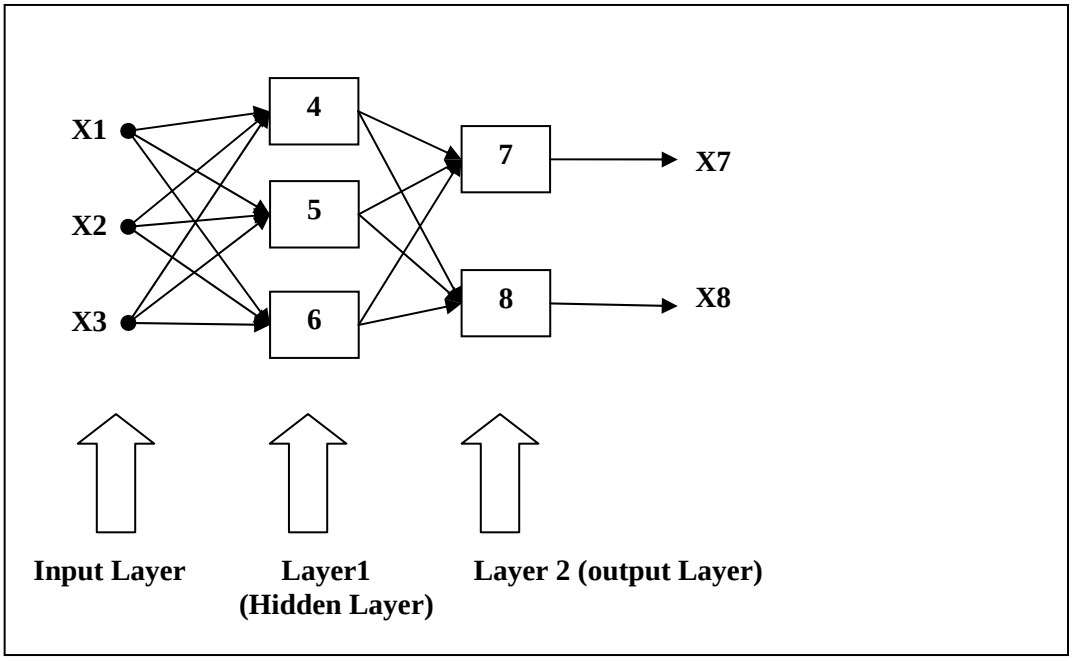


Figure 5.7 A 3-3-2 adaptive network.

□

5.3 Back-propagation learning rule

This section introduces a basic learning rule for adaptive networks. The central part of this learning rule concerns how to recursively obtain a gradient vector in which each element is defined as the derivative of an error measure with respect to a

parameter. This is done by means of the chain rule in elementary calculus. The procedure of finding a gradient vector in a network structure is generally referred to as **back-propagation** because the gradient vector is calculated in the direction opposite to flow of the output of each node. Once the gradient is obtained, a number of derivative based optimization and regression techniques are available for updating the parameters. In particular, if we estimate the gradient vector in a simple steepest descent method, the resulting learning paradigm is often referred to as the **back-propagation learning rule**. We shall introduce this learning rule in the rest of this section.

Supposed that a given feed-forward adaptive network in layered representation has L layers and layer l (l=0, 1, …, L; l=0 represent the input layer) has $N(l)$ nodes. Then the output and function of node i $[i = 1, ..., N(l)]$ in layer l can be represented as $x_{l,i}$ and $f_{l,i}$ respectively, as shown in figure 5.8.

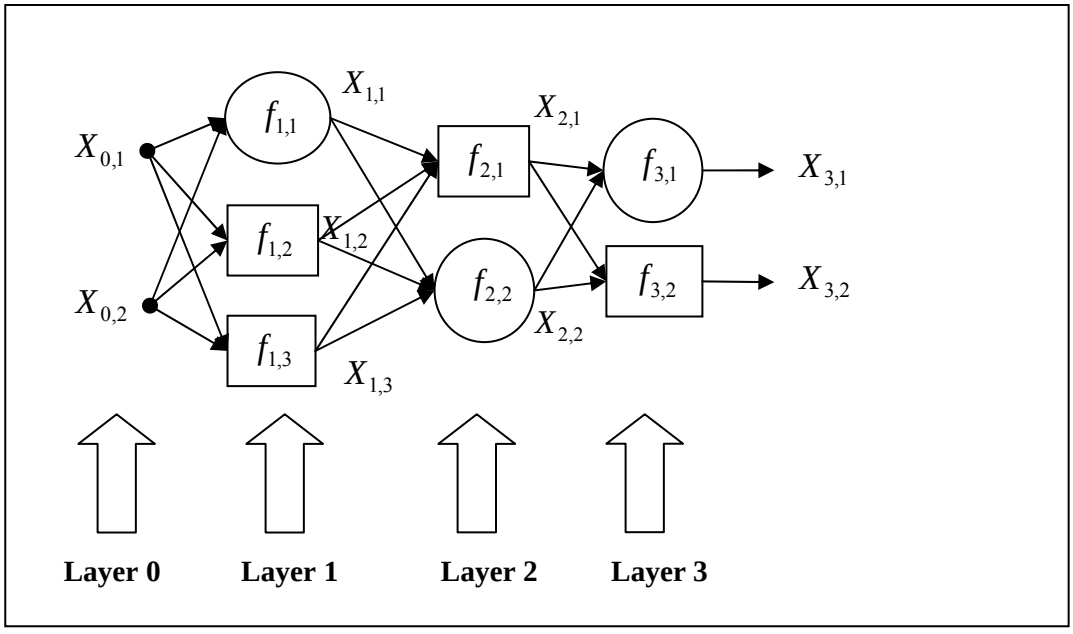


Figure 5.8 A layered representation of our notational conventions.

Without loss of generality, we assume that there are no jumping links (that is, links connecting nonconsecutive layers). Since the output of a node depends on the incoming signals and the parameter set of the node, we have the following general expression for the node function $f_{l,i}$:

$$x_{l,i} = f_{l,i}\left(x_{l-1,1},\ldots x_{l-1,N(l-1)},\alpha,\beta,\gamma,\ldots\right),$$

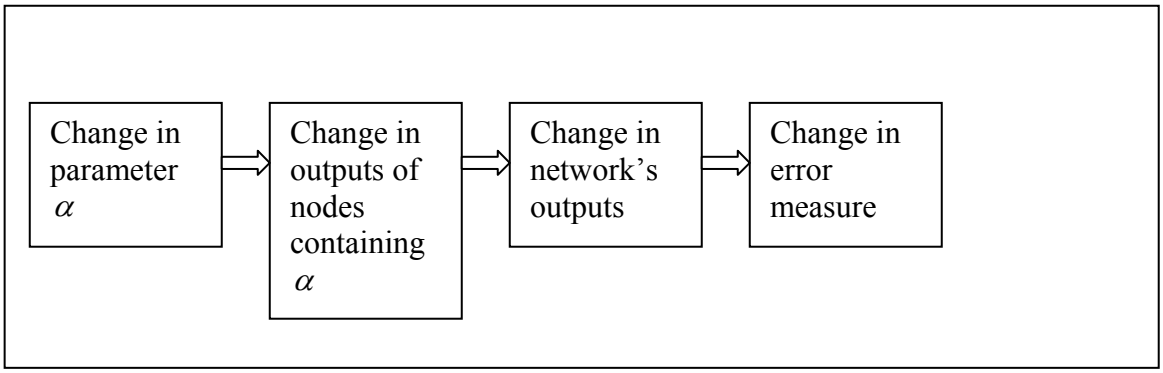
where $\alpha,\beta,\gamma,$ etc. are the parameters of this node.

Assume that the training data set has *P* entries; we can define an error measure for the p^{th} $\left(1 \le p \le P\right)$ entry of the training data as the sum of squared errors:

$$E_p = \sum_{k=1}^{N(L)} \left(y_k - x_{L,k}\right)^2,$$

where y_k is the k^{th} component of the p^{th} desired output vector and $x_{L,k}$ is the k^{th} component of the actual output vector produced by presenting p^{th} input vector to the network. (For notational simplicity, we omit the subscript *p* for both y_k and $x_{L,k}$.) Obviously, when E_p is equal to zero the network is able to reproduce exactly the desired output vector in the p^{th} training data pair. Thus our task here is to minimize an overall error measure, which is defined as $E = \sum_{p=1}^{P} E_p$.

To use steepest descent to minimize the error measure, first we have to obtain the gradient vector. Before calculating the gradient vector, we should observe the following causal relationships:



where the arrows indicate causal relationships. In other words, a change in a parameter α will affect the output of the node containing α ; this in turn will affect the output of the final layer and thus the error measure. Therefore, the basic concept in calculating the gradient vector is to pass a form of derivative information starting from the output layer and going backward layer by layer until the input layer is reached.

To facilitate the discussion, we define **error signal** $\varepsilon_{l,i}$ as the derivative of the error measure E_p with respect to the output of node i in layer l , taking both direct (where nodes at layer *l* and *l+1* share the same parameter) and indirect paths (where nodes at layer l and $l+n,\ n \in \mathbb{N} > 1$ effect by the same parameter) into consideration. In symbol,

$$\varepsilon_{l,i} = \frac{\partial^+ E_p}{\partial x_{l,i}}.$$

The expression was called the **ordered derivative** by Werbos[4]. The difference between the ordered derivative and the ordinary partial derivative lies in the way we view the function to be differentiated. For an internal nodes output $x_{l,i}$ (where $l \neq L$), the partial

derivative $\frac{\partial E_p}{\partial x_{l,i}}$ is equal to zero, since E_p does not depend on $x_{l,i}$ directly. However, it is obvious that E_P does depend on $x_{l,i}$ indirectly, since a change in $x_{l,i}$ will propagate through indirect paths to the output layer and thus produce a corresponding change in the value of E_p. Therefore, $\varepsilon_{l,i}$ can be viewed as the ratio of these two changes. The following example demonstrates the difference between the ordered derivative and the ordinary partial derivative.

Example 5.5 Ordered derivatives and ordinary partial derivatives

Consider a simple adaptive network shown in Figure 5.9 where z is a function of x and y, and y is in turn a function of x:

$$z = g(x, y), y = f(x),$$

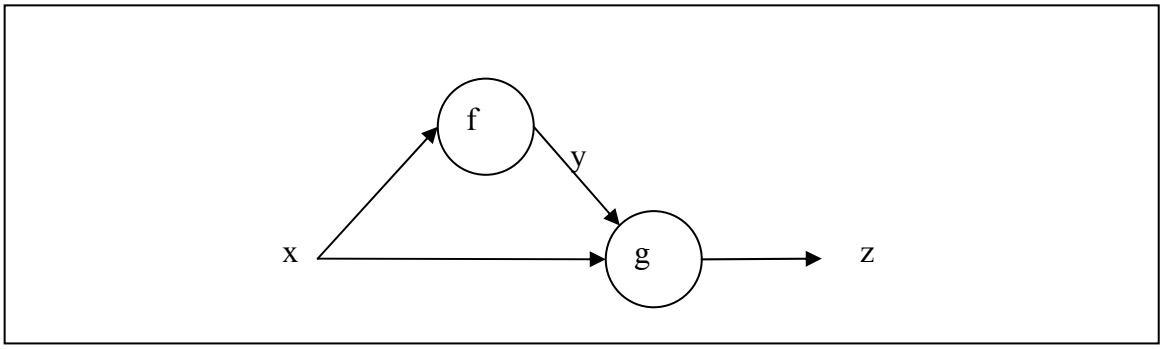


Figure 5.9 Ordered derivatives and ordinary partial derivatives example

then for the ordinary partial derivative $\frac{\partial z}{\partial x}$ where all the other input variables are assume constant is:

$$\frac{\partial z}{\partial x} = \frac{\partial g(x, y)}{\partial x},$$

but for ordered derivative, we take indirect causal relationship into consideration:

$$\frac{\partial^{+} z}{\partial x} = \frac{\partial g(x, f(x))}{\partial x} = \frac{\partial g(x, y)}{\partial x}\bigg|_{y=f(x)} + \frac{\partial g(x, y)}{\partial y}\bigg|_{y=f(x)} \frac{\partial f(x)}{\partial x}.$$

Therefore, the ordered derivative takes into consideration both direct and indirect paths that lead to the causal relationship.

Here is a specific example. Let's assume that

$$z = y^2 + x^2,\ y = x^2.$$

Then the ordinary partial derivative will be

$$\begin{aligned}\frac{\partial z}{\partial x} &= \left(y^2 + x^2\right)\frac{\partial}{\partial x} \\ &= y^2 + 2x.\end{aligned}$$

Whereas the order derivative will be

$$\begin{aligned}\frac{\partial^{+} z}{\partial x} &= \left(x^4 + x^2\right)\frac{\partial}{\partial x} + \left(\left(y^2 + y\right)\frac{\partial}{\partial y}\right)x^2\frac{\partial}{\partial x}. \\ &= 4x^3 + 4x + 4xy.\end{aligned}$$

□

The error signal for the i^{th} output at the last layer (layer L) has no indirect causal relationship it can be calculate directly:

$$\varepsilon_{L,i} = \frac{\partial^+ E_p}{\partial x_{L,i}} = \frac{\partial E_p}{\partial x_{L,i}} = -2\left(y_{L,i} - x_{L,i}\right). \tag{5.1}$$

For the internal node at the i^{th} position of layer l, the error signal is:

$$\varepsilon_{l,\,i} = \sum_{m=1}^{N(l+1)} \frac{\partial^+ E_p}{\partial x_{l+1,\,m}} \frac{\partial f_{l+1,\,m}}{\partial x_{l,\,i}} = \sum_{m=1}^{N(l+1)} \varepsilon_{l+1,\,m} \frac{\partial f_{l+1,\,m}}{\partial x_{l,\,i}} \tag{5.2}$$

where $0 \le l \le L-1$. That is, the error signal of an internal node at layer l can be expressed as a linear combination of the error signal of the nodes at layer l+1. Therefore, for any l and i $\left[0 \le l \le L \text{ and } 1 \le i \le N(l)\right]$, we can find $\varepsilon_{l,i} = \dfrac{\partial^+ E_p}{x_{l,i}}$ by first applying Equation (5.1) once to get error signals at the output layer, and then applying Equation (5.2) iteratively until we reach the desired layer l. The underlying procedure is called back-propagation since the error signals are obtained sequentially from the, output layer back to the input layer. The following is an example of how the error signal is calculated.

Example 5.6 Adaptive network and its error-propagation.

Figure 5.10 is an adaptive network, where each node is indexed by a unique number.

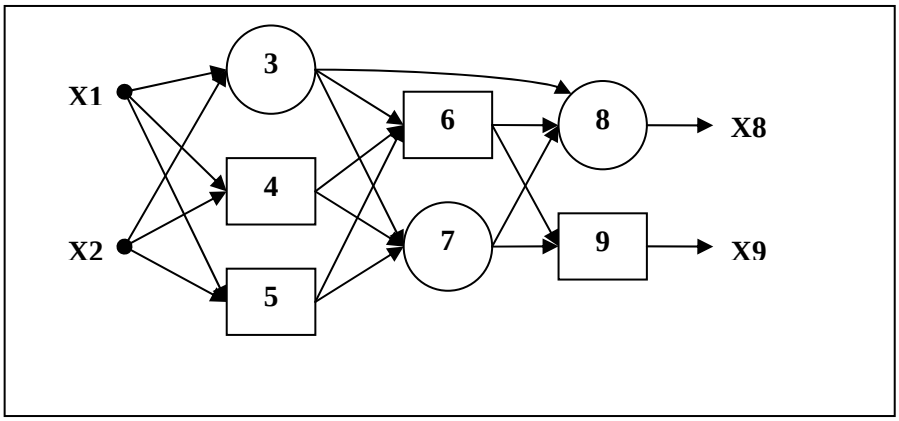


Figure 5.10 The adaptive network in example 5.6.

Again we use f_i and x_i to denote the function and output of node *i*. In symbols, if we choose the square error measure for E_p, then we have the following:

$$\varepsilon_9 = \frac{\partial^+ E_p}{\partial x_9} = \frac{\partial E_p}{\partial x_9} = -2\left(y_9 - x_9\right),$$

$$\varepsilon_8 = \frac{\partial^+ E_p}{\partial x_8} = \frac{\partial E_p}{\partial x_8} = -2\left(y_8 - x_8\right),$$

$$\varepsilon_7 = \frac{\partial^+ E_p}{\partial x_7} = \frac{\partial^+ E_p}{\partial x_8}\frac{\partial f_8}{\partial x_7} + \frac{\partial^+ E_p}{\partial x_9}\frac{\partial f_9}{\partial x_7} = \varepsilon_8 \frac{\partial f_8}{\partial x_7} + \varepsilon_9 \frac{\partial f_9}{\partial x_7},$$

$$\varepsilon_6 = \frac{\partial^+ E_p}{\partial x_6} = \frac{\partial^+ E_p}{\partial x_8}\frac{\partial f_8}{\partial x_6} + \frac{\partial^+ E_p}{\partial x_9}\frac{\partial f_9}{\partial x_6} = \varepsilon_8 \frac{\partial f_8}{\partial x_6} + \varepsilon_9 \frac{\partial f_9}{\partial x_6}.$$

Thus nodes 9 and 8 in the back-propagation network are only buffer nodes. Similar expressions can be written for the error signals of nodes 1, 2, 3, 4, and 5.

□

The gradient vector is defined as the derivative of the error measure with respect to each parameter. So we have to apply the chain rule again to find the gradient vector. If α is a parameter of the i^{th} node at layer *l* then

$$\frac{\partial^+ E_p}{\partial \alpha} = \frac{\partial^+ E_p}{\partial x_{1,i}} \frac{\partial x_{1,i}}{\partial \alpha} = \varepsilon_{1,i} \frac{\partial x_{1,i}}{\partial \alpha}. \qquad (5.3)$$

Note that if we allow the parameter α to be shared between different nodes, then Equation (5.3) should be changed to a more general form:

$$\frac{\partial^+ E_p}{\partial \alpha} = \sum_{x^* \in S} \frac{\partial^+ E_p}{\partial x^*} \frac{\partial f^*}{\partial \alpha}. \qquad (5.4)$$

where *S* is the set of nodes containing α as a parameter; and x^* and f^* are the output and function, respectively, of a generic node in *S*. The derivative of the overall error measure *E* with respect to α is

$$\frac{\partial^+ E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial^+ E_p}{\partial \alpha}, \qquad (5.5)$$

Accordingly the update formula for the generic parameter α is

$$\Delta \alpha = -\eta \frac{\partial^+ E}{\partial \alpha}, \qquad (5.6)$$

in which η is the **learning rate**, which can be express as

$$\eta = \frac{\kappa}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}}, \qquad (5.7)$$

where κ is the **step size**, the length of each transition along the gradient direction in the parameter space. If κ is small, the gradient method will closely approximate the gradient path, but the convergence will be slow since gradient must be calculate many times. On the other hand, if κ is large, convergence will initially be very fast, but the algorithm will oscillate about the optimum. Based on this observation two heuristic rules are often used:

1. If the error measure undergoes 4 consecutive reductions, increase κ by 10%.
2. If the error measure undergoes 2 consecutive combinations of 1 increase and 1 reduction, decrease κ by 10%.

The back-propagation learning rule can be improved by combining with the least square estimator. This learning rule is often called the **Hybrid learning rule,** which will be introduced in the next section.

5.4 Hybrid learning rules: Combining Back-propagation and Least square estimator

Although we can apply back-propagation to identify the parameters in an adaptive network, this optimization method usually takes a long time before it converges. We may observe, however, that an adaptive network's output (assuming there is only one) is linear in some of the network's parameters; thus we can identify these linear parameters by the linear least-squares estimator (LSE). This approach leads to a hybrid learning rule [1, 23] which combines the back-propagation learning rule and least-squares estimator for fast identification of parameters. The following is the mechanism of the hybrid learning rule.

Let’s assume that the adaptive network under consideration has only one output

represented by

$$out = F(\mathrm{i}, S), \qquad (5.8)$$

where **i** is the vector of input variables, S is the set of parameters, and F is the overall function implemented by the adaptive network. If there exists a function H such that the composite function $H \circ F$ is linear in some of the elements of S, then these elements can be identified by the least-squares method. More formally, if parameter set S can be divided into two sets

$$S = S_1 \oplus S_2 \qquad (5.9)$$

(where $\oplus$ represents direct sum) such that $H \circ F$ is linear in the elements of S_2, then upon applying H to Equation (5.9), we have

$$H(out) = H \circ F(\mathrm{i}, S), \qquad (5.10)$$

which is linear in the elements of S_2. Now given values of elements of S_1, we plug P training data into Equation (5.10) and obtain a matrix equation:

$$A\theta = Y \qquad (5.11)$$

where θ is an unknown vector whose elements are parameters in S_2. This is a standard linear least-squares problem and the best solution for θ, which minimizes $\|A\theta - y\|^2$, is the least-squares estimator (LSE) θ^*:

$$\theta^* = \left(A^T A\right)^{-1} A^T y, \qquad (5.12)$$

where A^T is the transpose of A and $\left(A^T A\right)^{-1} A^T$ is the pseudo-inverse of A if A is nonsingular.

Now we can combine back-propagation learning rule and the least-squares estimator to update the parameters in an adaptive network. For hybrid learning to be applied each epoch is composed of a **forward pass** and a **backward pass.** In the forward pass, after an input vector is presented, we calculate the node outputs in the network layer by layer until a corresponding row in the matrices A and y in Equation (5.11) is obtained. This process is repeated for all the training data pairs to form the complete A and y; then parameters in S_2 are identified by the pseudo-inverse formula in Equation (5.12).

After the parameters in S_2 are identified, we can compute the error measure for each training data pair. In the backward pass, the error signals [see equations (5.1) and (5.2)] propagate from the output end toward the input end; the gradient vector is accumulated for each training data entry. At the end of the backward pass for all training data, the parameters in S_2, are updated by steepest descent in Equation (5.6).

For given fixed values of the parameters in S_1, the parameters in S_2 thus found are guaranteed to be the global optimum point in the S_2 parameter space because of the choice of the squared error measure. Not only can this hybrid learning rule decrease the dimension of the search space explored by the original back-propagation learning rule, but, in general, it will also substantially reduce the time needed to reach convergence.

Chapter 6

6 Adaptive Neuro-Fuzzy Inference Systems

6.1 Introduction

The architectures and learning rules of adaptive networks have been described in the previous chapter. Functionally, there are almost no constraints on the node functions of an adaptive network except for the requirement of piecewise differentiability. Structurally, the only limitation on the network configuration is that it should be of the feed-forward type if we do not want to use more complex asynchronously operated model. Because of these minimal restrictions, adaptive networks can be employed directly in a wide variety of applications of modeling, decision making, signal processing, and control.

In this chapter, we introduce a class of adaptive networks that are functionally equivalent to fuzzy inference systems. The architecture is referred to as ANFIS, which stands for **Adaptive Neuro-Fuzzy Inference System** which has been proposed by Jang [1]. Next we describe how to decompose the parameter set to facilitate the hybrid learning rule for ANFIS architectures representing the Mamdani, Sugeno and Tsukamoto fuzzy models.

Note that similar network structures were also proposed independently by Lin and Lee [26] and Wang and Mendel [27].

6.2 ANFIS Architecture

Let us consider a fuzzy system, with two if-then rules of Sugeno type:

Rule *i*: if *x* is A_i and *y* is B_i then $z = p_i x + q_i y + r_i$, i=1,2.

Corresponding ANFIS architecture for this system is shown in Figure 6.1.

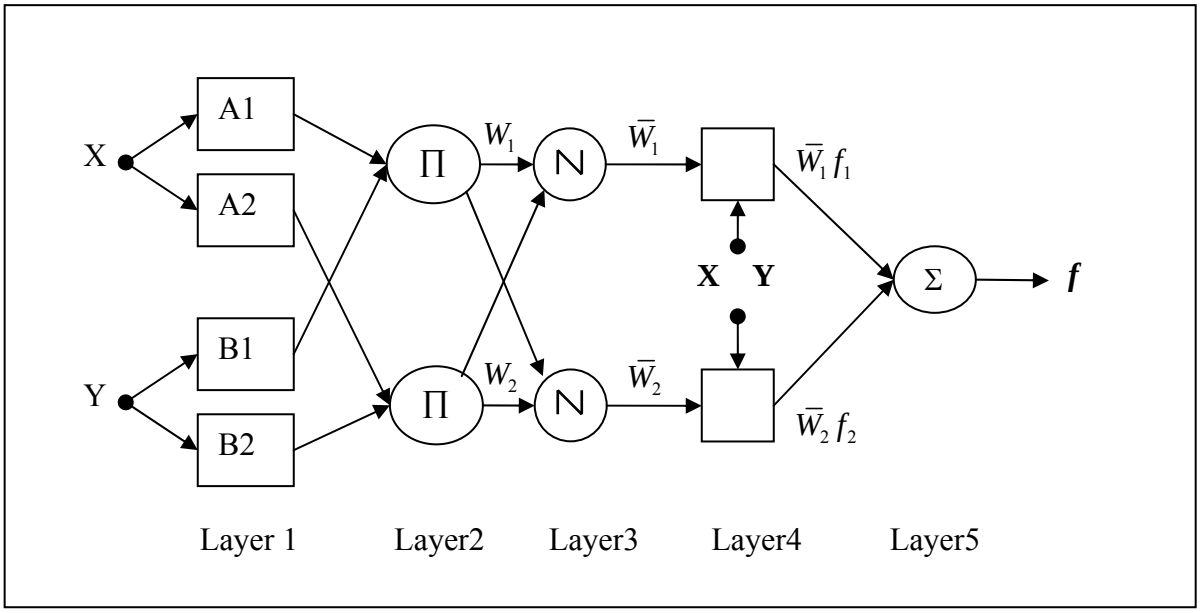


Figure 6.1 ANFIS architecture for Sugeno type reasoning.

The node functions in the same layer are of the same function family, as described below.

Layer 1: The output of the i^{th} node of this layer is

$$o_i^1 = \mu_{A_i}(x) \quad \text{for } i = 1,2 \text{ or}$$
$$o_i^1 = \mu_{B_{i-2}}(x) \text{ for } i = 3,4,$$

where *x* is the input to the i^{th} node and A_i (or B_{i-2}) is the linguistic label associated with this node function. Generally the bell-shaped membership function is used such as:

$$\mu_{*_i}(x) = \frac{1}{1 + \left| \frac{x - c_i}{a_i} \right|^{2b_i}}$$

where $\{a_i, b_i, c_i\}$ are the parameters which are referred to as the **premise parameter** and * is the linguistic label *A* or *B*. As the values of these parameters change, the bell-shaped function varies accordingly, thus exhibiting various forms of membership functions for fuzzy sets *A* and *B*.

Layer 2: Nodes of this layer multiply together the incoming signals and send the product out. Each node output represents the firing strength w_i of the corresponding rule where

$$w_i = \mu_{A_i}(x)\,\mu_{B_i}(y), \quad i = 1, 2.$$

In general, any other T-norm operators (Section 2.4.1) that perform fuzzy AND can be used as the node function of this layer.

Layer 3: The outputs of the nodes in this layer are normalized firing strengths $\overline{w}_i$ where

$$\overline{w}_i = \frac{w_i}{w_1 + w_2}, \; i = 1, 2.$$

For convenience, the outputs of this layer are called **normalised firing strength**.

Layer 4: The output of the i^{th} node of this layer is

$$o_i^4 = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i),$$

where $\{p, q, r\}$ are the parameters which are called the **consequent parameter**.

Layer 5: The single node in this layer computes the overall system output as the summation of all incoming signals:

$$o^5 = \sum_i \overline{w}_i f_i .$$

Thus we have constructed an adaptive network that is functionally equivalent to a Sugeno fuzzy model. Note that the structure of this adaptive network is not unique. We can combine layers 3 and 4 to obtain an equivalent network with only four layers. In the extreme case, we can even shrink the whole network into a single adaptive node with the same parameter set. Obviously, the assignment of node functions and the network configuration are arbitrary, as long as each node and each layer perform meaningful and modular functionalities.

Generally Sugeno ANFIS is often simply called “ANFIS” because it has been the choice of its author however several other models have been tried namely the Tsukamoto ANFIS and Mamdani ANFIS. The modification from Sugeno ANFIS to Tsukamoto ANFIS is straightforward, as shown in Figure 6.2, where the output of each rule (f_i, i = 1, 2) is induced jointly by a consequent membership function and a firing strength.

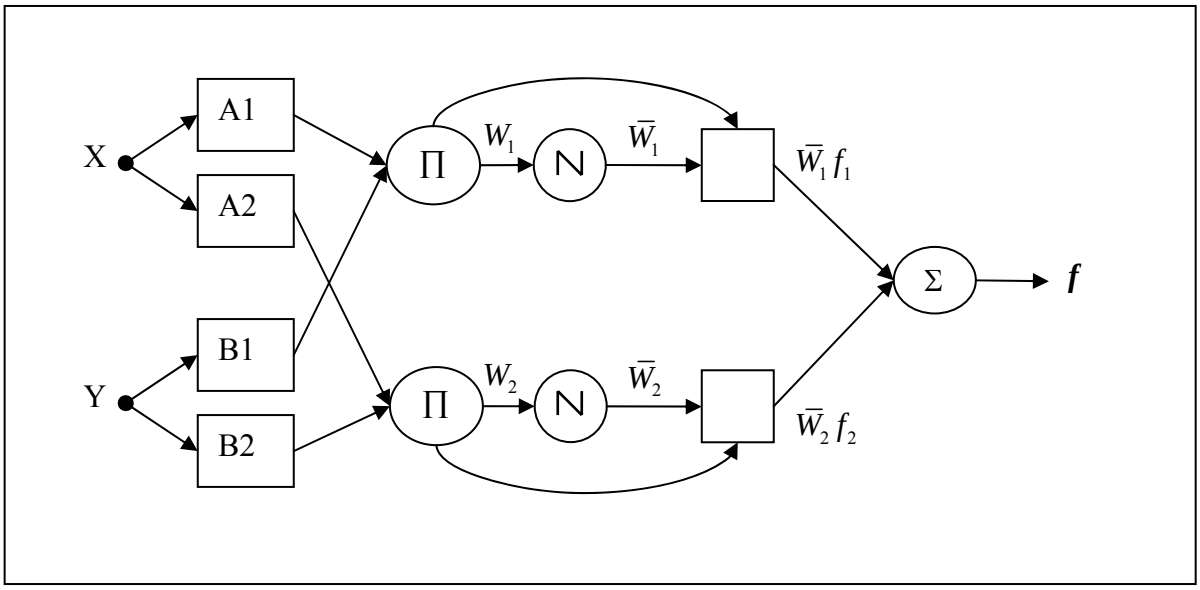


Figure 6.2 ANFIS architecture of two inputs two rules Tsukamoto fuzzy model.

For the Mamdani fuzzy inference system with max-min composition, a corresponding ANFIS can be constructed if discrete approximations are used to replace the integrals in the centroid defuzzification scheme introduced in Section 4.2. However, the resulting ANFIS is much more complicated than either Sugeno ANFIS or Tsukamoto ANFIS. The extra complexity in structure and computation of Mamdani ANFIS with max-min composition does not necessarily imply better learning capability or approximation power. If we adopt sum-product composition and centroid defuzzification for a Mamdani fuzzy model, a corresponding ANFIS can be constructed easily based on Theorem 4.1 directly without using any approximations at all. Although ANFIS can be used with several fuzzy inference systems, the author of ANFIS has chosen to use the first order Sugeno fuzzy model in the system for its transparency and efficiency [1]. Therefore from this point onward, if not mentioned otherwise, the first order Sugeno ANFIS will be called ANFIS.

6.3 Hybrid Learning Algorithm

From the ANFIS architecture example shown in Figure 6.1, we observe that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters. In symbols, the output f in Figure 6.1 can be rewritten as

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \bar{w}_1 (p_1 x + q_1 y + r_1) + \bar{w}_2 (p_2 x + q_2 y + r_2) \\ &= (\bar{w}_1 x) p_1 + (\bar{w}_1 y) q_1 + (\bar{w}_1) r_1 + (\bar{w}_2 x) p_2 + (\bar{w}_2 y) q_2 + (\bar{w}_2) r_2, \end{aligned}$$

which is linear in the consequent parameters p_1, q_1, r_1, p_2, q_2 and r_2. From this observation, we have *S* represent set of total parameters $\{\bar{w}_1, \bar{w}_2, p_1, q_1, r_1, p_2, q_2, r_2\}$, *S1* represent set of premise (nonlinear) parameters $\{\bar{w}_1, \bar{w}_2\}$, *S2* represent set of consequent (linear) parameters $\{p_1, q_1, r_1, p_2, q_2, r_2\}$ in Equation (5.9); and $H(\cdot)$ and $F(\cdot, \cdot)$ are the identity function and the function of the fuzzy inference system, respectively, in Equation (5.10). Therefore, the hybrid learning algorithm developed in Section 5.4 can be applied directly. More specifically, in the forward pass of the hybrid learning algorithm, node outputs go forward until layer 4 and the consequent parameters are identified by the least-squares method. In the backward pass, the error signals propagate backward and the premise parameters are updated by gradient descent. Table 6.1 summarizes the activities in each pass.

	Forward pass	**Backward pass**
Premise parameters	Fixed	Gradient descent
Consequent parameters	Least square estimator	Fixed
Signals	Node outputs	Error signals

Table 6.1 Two passed in the hybrid learning procedure for ANFIS

As mentioned in Section 5.4, the consequent parameters thus identified are optimal under the condition that the premise parameters are fixed. Accordingly, the hybrid approach converges much faster since it reduces the search space dimensions of the original pure back-propagation method [1]. Thus we should always look for the possibility of decomposing the parameter set in the first place. The following example will show the mechanism of the hybrid learning procedure in ANFIS.

Example 6.1 Hybrid learning procedure for ANFIS

For the forward pass let N be the total number of inputs in ANFIS architect, R is the number of rules for each input, $T = R^N$ is the number of nodes in layer 1. We estimate the consequence parameters using LSE by rewriting output of the layer 5 as

$$A\Theta = \begin{bmatrix} \overline{w}_1 x_1 \\ \overline{w}_1 x_2 \\ \vdots \\ \overline{w}_1 x_N \\ \overline{w}_1 \\ \vdots \\ \overline{w}_T x_1 \\ \overline{w}_T x_2 \\ \vdots \\ \overline{w}_T x_N \\ \overline{w}_T \end{bmatrix} \begin{bmatrix} \theta_{1,1} \\ \theta_{2,1} \\ \vdots \\ \theta_{N,1} \\ \theta_{(N+1),1} \\ \vdots \\ \theta_{1,T} \\ \theta_{2,T} \\ \vdots \\ \theta_{N,T} \\ \theta_{(N+1),T} \end{bmatrix}$$

The estimation of consequence parameters is $\Theta^* = (A^T A)^{-1} A^T f$, where f is the output of layer 5.

For backward pass let's assume that the consequence parameters are fixed, the error rate for premise parameters can be calculated as follows (recall equations 5.2 and 5.3 in previous chapter):

$$\varepsilon_{l,\,i} = \sum_{m=1}^{N(l+1)} \varepsilon_{l+1,\,m} \frac{\partial o_{l+1,\,m}}{\partial o_{l,\,i}}$$

$$\frac{\partial^{+} E_p}{\partial \beta} = \varepsilon_{\,1,\,i} \frac{\partial o_{1,\,i}}{\partial \beta}.$$

where $\varepsilon_{l,\,i}$ is the error signal of the i^{th} node at layer l, β is the premise parameter, l=0, 1, …, 5; has $N(l)$ nodes.

Also let's define the measure of error for the k^{th} learning pattern in the same way, as in common back-propagation networks:

$$E_k = \frac{1}{2}\left(y_k - o_k\right)^2$$

Then the solution of $\frac{\partial E}{\partial o_5}$ is $-\left(y - o\right)$. The derivation of $\frac{\partial o_{j,i}}{\partial o_{j-1,m}}$, $j = 5,...,2$, i=1,…, *N(j)* and m= 1,…, *N(j-1)* are

$$\frac{\partial o_{5,i}}{\partial o_{4,m}} = \frac{\partial\left(\sum_{m=1}^{N(4)} f_m \overline{w}_m\right)}{\partial\left(f_m \overline{w}_m\right)} = 1,$$

$$\frac{\partial o_{4,i}}{\partial o_{3,m}} = \frac{\partial\left(f_i \overline{w}_i\right)}{\partial\left(\overline{w}_i\right)} = f_i,$$

where i= m otherwise $\frac{\partial o_{4,i}}{\partial o_{3,m}} = 0$.

$$\frac{\partial o_{3,i}}{\partial o_{2,m}} = \frac{\partial}{\partial w_m}\left(\frac{w_i}{\sum_{k=1}^{N(2)} w_k}\right) = \frac{\sum_{k=1}^{N(2)} w_k - w_i}{\left(\sum_{k=1}^{N(2)} w_k\right)^2}$$

where $i = m$ otherwise $\frac{\partial o_{3,i}}{\partial o_{2,m}} = -\frac{w_i}{\left(\sum_{k=1}^{N(2)} w_k\right)^2}$.

$$\frac{\partial o_{2,\,i}}{\partial o_{1,m}} = \frac{\partial}{\partial A_m}\left(\prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j\right),$$

where $A_j \in \Re(A_m)$ denotes the fuzzy sets, which make the premise part of the rule containing fuzzy set A_m.

Using equations 5.3 and 5.4 we can calculate the derivative of the overall error measure at the p^{th} training set E_p with respect to β

$$\frac{\partial^+ E_p}{\partial \beta} = -(y - o)\frac{1}{\left(\sum_{j=1}^{N(2)} w_j\right)^2}\left(\sum_{k=1}^{N(2)}\sum_{j=1}^{N(2)}(f_k - f_j)w_j\right)\prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j \frac{\partial o_1}{\partial \beta}$$

Since bell membership function is used in ANFIS, β is referred by the three premised parameters $\{a_{ij}, b_{ij}, c_{ij}\}$ where

$$\frac{\partial o_1}{\partial a_{ij}} = \begin{cases} \dfrac{2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}\left(x-c_{ij}\right)}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}^2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} > 0 \\ \dfrac{-2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}\left(x-c_{ij}\right)}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}^2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} < 0 \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0 \end{cases}$$

$$\frac{\partial o_1}{\partial b_{ij}} = \begin{cases} \dfrac{-2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}} \ln\left(\left|\dfrac{x-c_{ij}}{a_{ij}}\right|\right)}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} \neq 0 \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0 \end{cases},$$

$$\frac{\partial o_1}{\partial c_{ij}} = \begin{cases} \dfrac{2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} > 0. \\ \dfrac{-2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} < 0. \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0. \end{cases}$$

where *i* is the number of the corresponding rule and j is the number of the corresponding linguistic variable in the rule. Then the update value of each premise parameter can be calculated by simply using equation 5.6 to 5.8 in the previous chapter.

□

Even though the hybrid learning algorithm reduces the search space dimensions of the original pure back-propagation method [1] thus it was chosen for ANFIS. There are several short coming in the algorithm.

First the forward pass employs the least squares estimator that relies on the calculation of the pseudo inverse $(A^T A)^{-1} A^T$. The pseudo inverse is guaranteed to exist for any full-rank matrix *A*. However, in some cases the matrix $A^T A$ is ill-conditioned; this occurs when the measurements are only marginally related to the estimated parameters. In these cases, the least squares estimate amplifies the measurement noise, and may be grossly inaccurate. This may occur even when the pseudo inverse itself can be accurately calculated numerically.

Another drawback of the least squares estimator is the fact that it seeks to minimize the norm of the measurement error $A\theta - f$. In many cases, one is truly interested in obtaining small error in the parameter θ , e.g., a small value of $\left\|\theta - \theta^*\right\|$. However, since θ is unknown, this quantity cannot be directly minimized.

Also better estimators can be constructed, an effect known as Stein's phenomenon [28, 29]. For example, if the measurement error is Gaussian, several estimators are known which dominate, or outperform, the least squares technique; the most common of these is the James-Stein estimator [28, 29].

Moreover the back-propagation learning rule in backward pass is highly complex due to the fact that the ordered derivative has to be calculated for every node and since the bell membership function contains absolute value, which is necessary otherwise the derivative of the membership function will not work when the input exceeds the centre of the membership function. Therefore, the derivative for each parameter in the membership function varies according to different conditions thus adding more complexity to the system. All of these complexities will eventually results in slow convergence of the consequence parameters.

Therefore, we shall introduce a new learning mechanism for ANFIS in the following chapter.

Chapter 7

7 Faster Adaptive Neuro-Fuzzy Inference Systems

7.1 Introduction

As we have mentioned in the previous chapter, the hybrid learning algorithm in ANFIS have several short coming mainly due to the insufficiency of least square estimate and the complexity of the ordered derivative. Here we will introduce a new learning algorithm that does not rely on the least square estimate for forward pass learning and reduce complexity of the ordered derivative in backward pass learning.

Furthermore, the results in this chapter will show that it is possible to reduce the computational cost if the proposed algorithm is used, due to the decrease in the number of iterations during the training process, as well as due to the fact that the algorithm allows substantial reduction in the amount of computational operations that have to perform during each single iteration.

7.2 New Learning Algorithm

In this algorithm the order derivatives are used only to determine the *direction* of the change of the parameters, but not the *size* of the change. (Riedmiller and Braun [6] points to a harmful and unforeseeable influence of the size of the partial derivative on the weight step.) Only the sign of the derivative is used to indicate the direction of the parameters update. This new learning algorithm is applicable to both premised and consequence parameters, therefore there is no need to use the least square estimate.

Let's assume α_t is a parameter we want to update at time t. The size of the parameter change $\Delta\alpha_t$ is determined by the so-called "update-value" Δ_t(7.1). The second step of

learning is to determine the new update-values Δ_t. This is based on a sign-dependent adaptation process (7.2).

$$\alpha_t = \begin{cases} -\Delta_t, \text{ if } \dfrac{\partial^+ E}{\partial \alpha_t} > 0, \\ +\Delta_t, \text{ if } \dfrac{\partial^+ E}{\partial \alpha_t} < 0, \\ 0, \text{ otherwise,} \end{cases} \tag{7.1}$$

$$\Delta_t = \begin{cases} \eta^+ \Delta_{t-1}, \text{ if } \dfrac{\partial^+ E}{\partial \alpha_{t-1}} \cdot \dfrac{\partial^+ E}{\partial \alpha_t} > 0, \\ \eta^- \Delta_{t-1}, \text{ if } \dfrac{\partial^+ E}{\partial \alpha_{t-1}} \cdot \dfrac{\partial^+ E}{\partial \alpha_t} < 0, \\ \Delta_{t-1}, \text{ otherwise,} \end{cases} \tag{7.2}$$

The adaptation rule works as follows. Every time the order derivative of the corresponding parameter α_t changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value Δ_t is decreased by the factor η^-. If the derivative retains its sign, the update-value is increased by the factor η^+ in order to accelerate convergence in shallow regions. Additionally, in case of the derivative is zero, there should be no adaptation in the succeeding learning step.

Therefore, it is possible to substantially simplify the obtained derivatives, as we can reduce them by the elements that do not influence the sign. For the premise parameters $\beta \in \{a_{ij}, b_{ij}, c_{ij}\}$ the ordered derivative can be simplified to

$$\operatorname{sign}\left(\frac{\partial^{+}E}{\partial\beta}\right)=-(y-o)\left(\sum_{k=1}^{N(2)}\sum_{j=1}^{N(2)}(f_k-f_j)w_j\right)\operatorname{sign}\left(\frac{\partial o_1}{\partial\beta}\right),$$

where the partial derivatives of the output functions with respect to their parameters are

$$\frac{\partial o_1}{\partial a_{ij}}=\begin{cases}\dfrac{2b_{ij}}{a_{ij}}y_{ij}\left(1-y_{ij}\right), & \text{if } x\neq c_{ij},\\ 0, & \text{if } x=c_{ij},\end{cases}$$

$$\frac{\partial o_1}{\partial b_{ij}}=\begin{cases}-2\ln\left|\dfrac{x-c_{ij}}{a_{ij}}\right|y_{ij}\left(1-y_{ij}\right), & \text{if } x\neq c_{ij},\\ 0, & \text{if } x=c_{ij},\end{cases}$$

$$\frac{\partial o_1}{\partial c_{ij}}=\begin{cases}\dfrac{2b_{ij}}{x-c_{ij}}y_{ij}\left(1-y_{ij}\right), & \text{if } x\neq c_{ij},\\ 0, & \text{if } x=c_{ij},\end{cases}$$

where

$$y_{ij}=\frac{1}{1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}}.$$

Since we only need the sign of the derivative these can be further simplify to

$$\operatorname{sign}\left(\frac{\partial o_1}{\partial a_{ij}}\right)=\frac{b_{ij}}{a_{ij}},$$

$$\operatorname{sign}\left(\frac{\partial o_1}{\partial b_{ij}}\right)=\begin{cases}-\ln\left|\dfrac{x-c_{ij}}{a_{ij}}\right|, & \text{if } x\neq c_{ij}\\ 0, & \text{otherwise}\end{cases}$$

$$\operatorname{sign}\left(\frac{\partial o_1}{\partial c_{ij}}\right)=\begin{cases}\dfrac{b_{ij}}{x-c_{ij}}, & \text{if } x\neq c_{ij}\\ 0, & \text{otherwise}\end{cases}.$$

Similarly using the notation of Example 6.5 the order derivative of consequence parameters Θ are

$$\frac{\partial^{+}E}{\partial\Theta}=-(y-O)\frac{\partial o_4}{\partial\Theta},$$

where partial derivatives of the output function at the fourth layer with respect to the consequence parameters are

$$\frac{\partial o_4}{\partial\theta_{i,j}}=\frac{\partial}{\partial\theta_{i,j}}\begin{bmatrix}\overline{w}_1x_1\\ \overline{w}_1x_2\\ \vdots\\ \overline{w}_1x_N\\ \overline{w}_1\\ \vdots\\ \overline{w}_Tx_1\\ \overline{w}_Tx_2\\ \vdots\\ \overline{w}_Tx_N\\ \overline{w}_T\end{bmatrix}\begin{bmatrix}\theta_{1,1}\\ \theta_{2,1}\\ \vdots\\ \theta_{N,1}\\ \theta_{(N+1),1}\\ \vdots\\ \theta_{1,T}\\ \theta_{2,T}\\ \vdots\\ \theta_{N,T}\\ \theta_{(N+1),T}\end{bmatrix}=\overline{w}_jx_i$$

where $i=1,...,N-1$; $j=1,...,T$; N = total number of inputs in ANFIS architect; $T=R^N$ where R = the number of rules for each input is the number of nodes in layer 1 and x_i = the input where $x_{N+1}=1$.

7.3 Modelling FANFIS

The following sections 7.3.1 to 7.3.4 are examples of FANFIS modelling in several applications such as chaotic time series predictions, modelling three-input nonlinear function, dynamical systems identifications and stock price predictions. All of the examples were simulated on a dual AMD Athlon ^{TM} MP2600+ with two gigabyte of memory running under Windows 2003 server with the parameters $\eta^- = 0.5$ and $\eta^+ = 1.2$ as empirically suggested in [46]. The choice of the decrease factor η^- and increase factor η^+ was lead by the following considerations: if a jump over a minimum occurred, the previous update value was too large. For it is not know from the gradient information how much the minimum was missed, in average it will be a good guess to halve the update value, i.e. $\eta^- = .5$. The increase factor η^+ has to be large enough to allow fast growth of the update value in shallow regions of the error function, on the other hand the learning process can be considerably disturbed, if a too large increase factor lead to persistent changes of the direction of the weight step, their for $\eta^+ = 1.2$ seems to be a good choice. The initial and final membership functions for each example are listed in the Appendix I.

7.3.1 Chaotic time series predictions

The chaotic time series used here is generated by the chaotic Mackey-Glass differential delay equation [5] defined as

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t).$$

The prediction values of this time series is a benchmark problem that has been used by a number of researchers such as Lapedes and Farber [30], Moody [31, 32], Jones et al. [33], Crowder [34] and Sanger [35]. The task is to use past values of the time series up to time t to predict the value at some point in the future $t+P$.

The standard method for this type of prediction is to create a mapping from *D* points of time series space Δ apart that is, $x\left(t-\left(D-1\right)\Delta\right),...,\ x\left(t-\Delta\right),\ x\left(t\right)$, to predict future value $x\left(t+P\right)$. To allow comparison with earlier work [9, 10, 11, 13], the value $D=4$ and $\Delta=P=6$ were used. The initial condition were $x\left(0\right)=1.2$ and $\tau=17$. In this way, $x\left(t\right)$ was obtained by numerical integration for $0\le t\le 2000$. From the Mackey-Glass time series $x\left(t\right)$, we extracted 1000 input-output data pairs of the following format:

$$\left[x\left(t-18\right),\ x\left(t-12\right),\ x\left(t-6\right),\ x\left(t\right);\ x\left(t+6\right)\right],$$

where $t=118$ to 1117. The first 500 pairs were used as the training set while the remaining 500 pairs were the testing data set. Two membership functions were assigned for each input so that we have 16 rules with a total of 104 parameters of which 24 are premise parameters and 80 are consequent parameters.

We compare the hybrid learning procedure of ANFIS with the FANFIS learning procedure by using the same number of epochs for each procedure. The hybrid procedure of ANFIS took about 171 seconds to complete 500 epochs while the new algorithm of FANFIS took only 98 seconds. The root mean square error (RMSE) for the training set and testing set are the same within five significant figures for both procedures. These are .0016 for the training set, and .0015 for the testing set as shown in Figures 7.1 and 7.2 which also shows faster and smoother convergence for the new procedure.

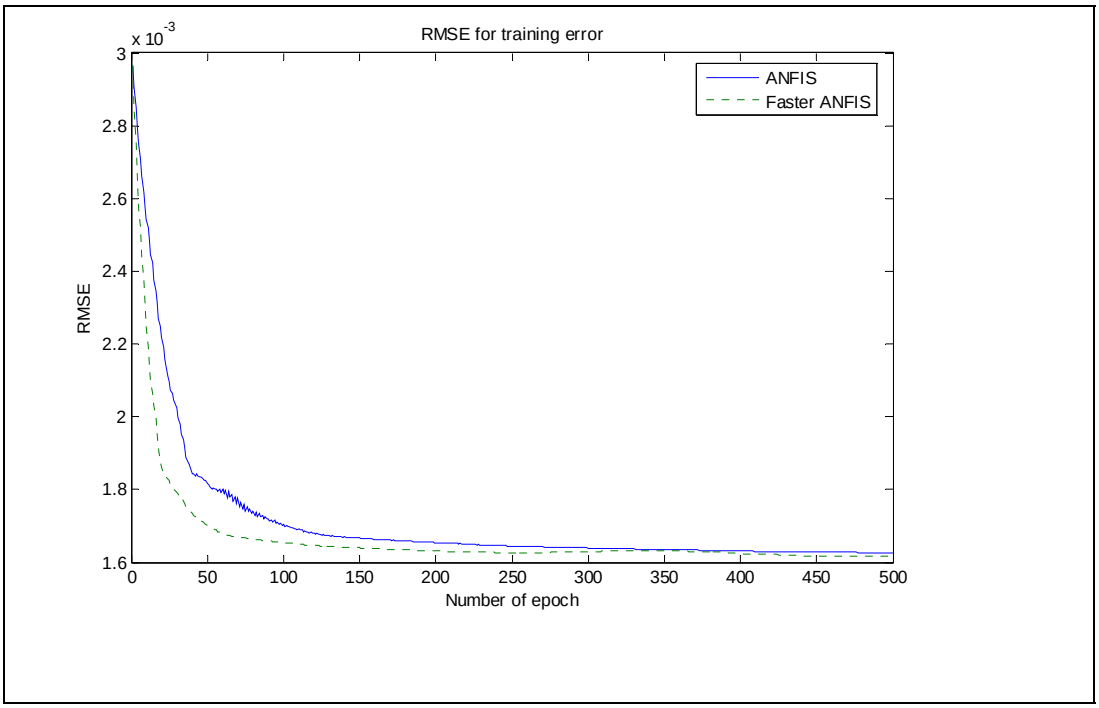


Figure 7.1 Training RMSE curves for ANFIS versus FANFIS.

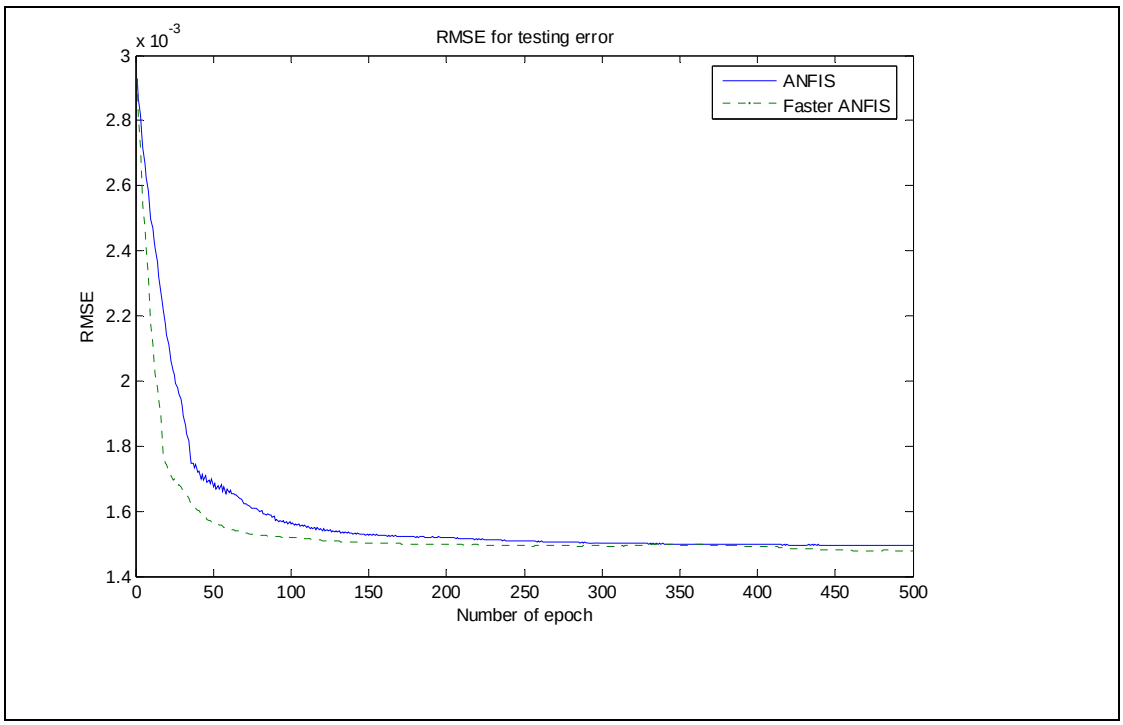


Figure 7.2 Testing RMSE curves for ANFIS versus FANFIS.

Since the RMSE of both procedures are relatively small the desired and the predicted values for both training data and testing data are essentially the same as shown in Figures 7.3 and 7.4. The differences between them can only been seen in much finer scale where the RMSE of prediction residual is 15575×10^{-7} for ANFIS and 15441×10^{-7} for FANFIS.

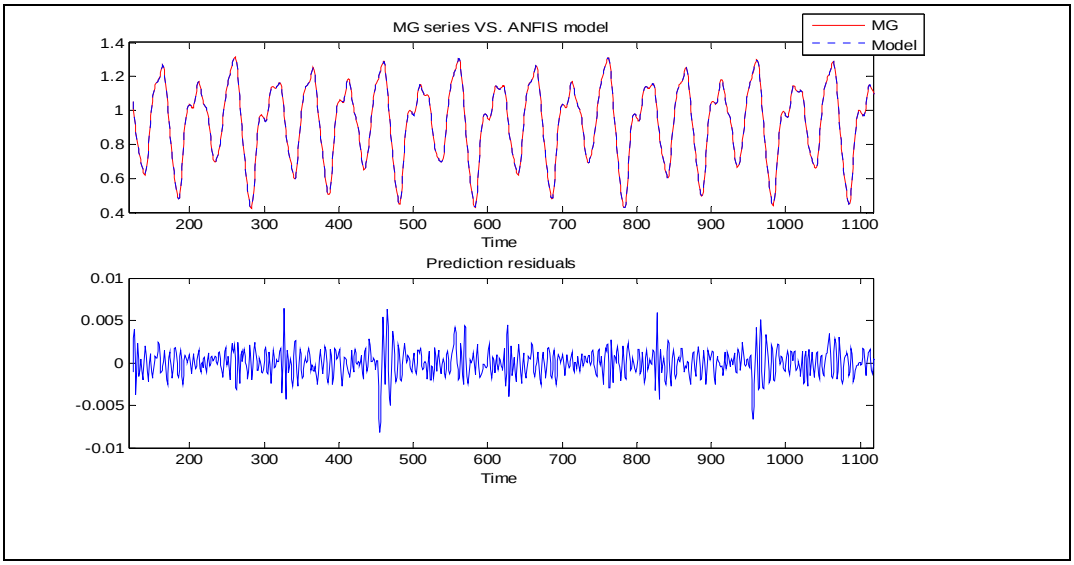


Figure 7.3 Mackey-Glass time series compare with ANFIS and its residuals.

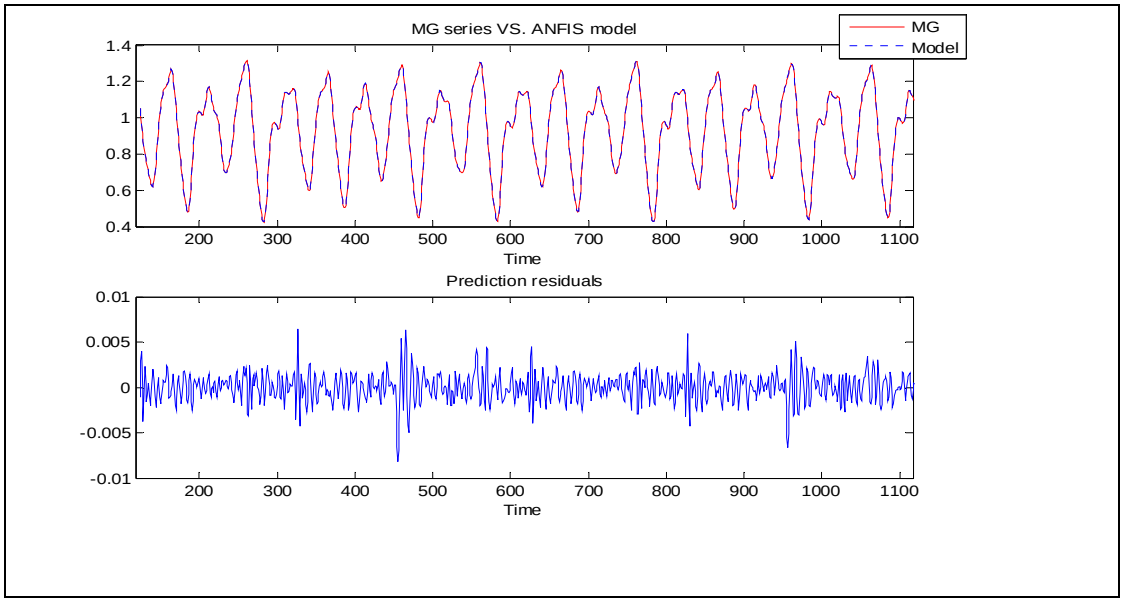


Figure 7.4 Comparison of Mackey-Glass time series and prediction using FANFIS and its residuals.

A better comparison for both models here is to show how long each model takes to reach the desired error. Figure 7.5 shows that FANFIS outperforms ANFIS in any given training error goals.

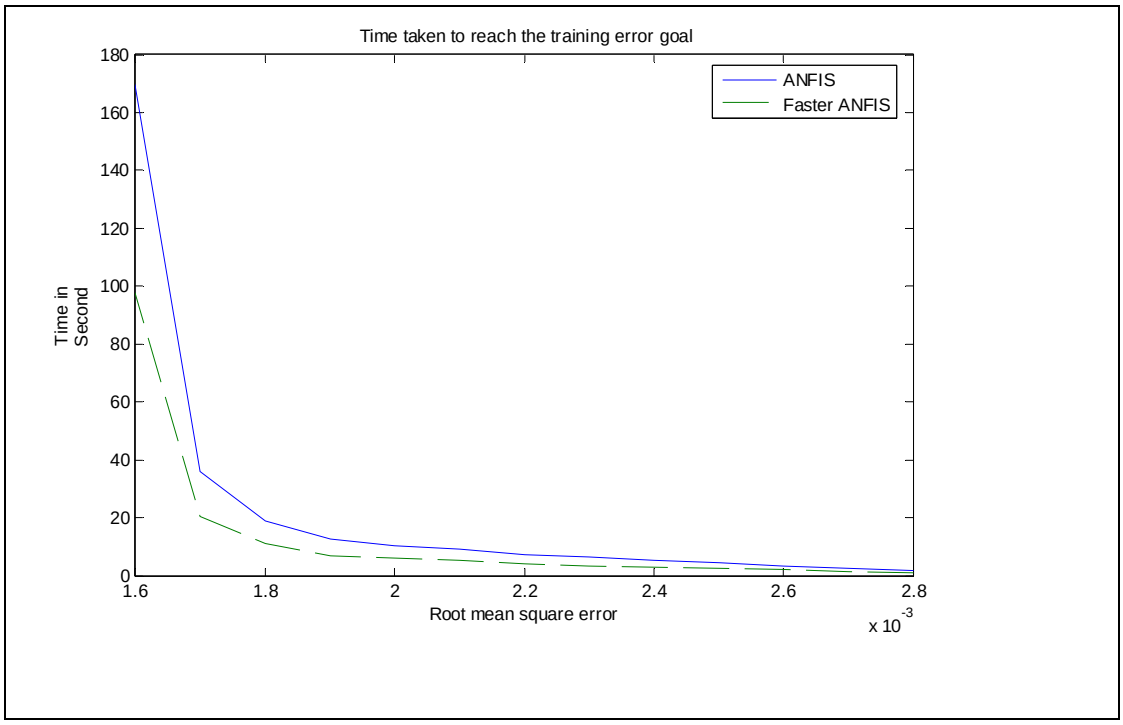


Figure 7.5 Time taken to reach a specific training error for ANFIS and FANFIS.

Table 7.1 lists the generalization capacities of other methods [34], which were measured by using each method to predict 500 points immediately following the training set. Here the non-dimensional error index (NDEI) [30, 34] is defined as the root mean square error divided by the standard deviation of the target series. (The last 4 rows are from [34].)

Method	NDEI	Time in second
FANFIS	0.0068543	98
ANFIS	0.0068944	171
Cascaded-correlation NN	0.06	N/A
Back-propagation MLP	0.02	N/A
6^{th} order polynomial	0.04	N/A
Linear predictive method	0.55	N/A

Table 7.1 Comparison of generalization capability

7.3.2 Modeling a Three-Input Nonlinear Function

The training data in this example were obtained from a three-input nonlinear equation defined by

$$output = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2,$$

This equation was also used by Takagi and Hayashi [36], Sugeno and Kang [37], and Kondo [38] to test their modeling approaches. Here the FANFIS architecture contains eight rules, with 2 membership functions assigned to each input variable.
A total of 216 training data and 125 testing data were sampled uniformly from the input range $[1,6]\times[1,6]\times[1,6]$ and $[1.5,5.5]\times[1.5,5.5]\times[1.5,5.5]$ respectively.

We compare FANFIS with ANFIS by using the same number of epochs for each procedure. The ANFIS took about 7.9 second to complete 200 epochs while FANFIS took only 3.6 second. The following figures will demonstrate that FANFIS outperforms ANFIS in term of errors and testing speed. Figures 7.6(a) and 7.6(b) show the training and testing error curves for FANFIS and ANFIS at each epoch. Here the error curve of FANFIS has a faster and smoother convergence where the final root mean square errors of training and testing are 0.0031 and 0.1977 respectively. In comparison, the final training and testing errors for ANFIS are 0.0069 and 0.2238.

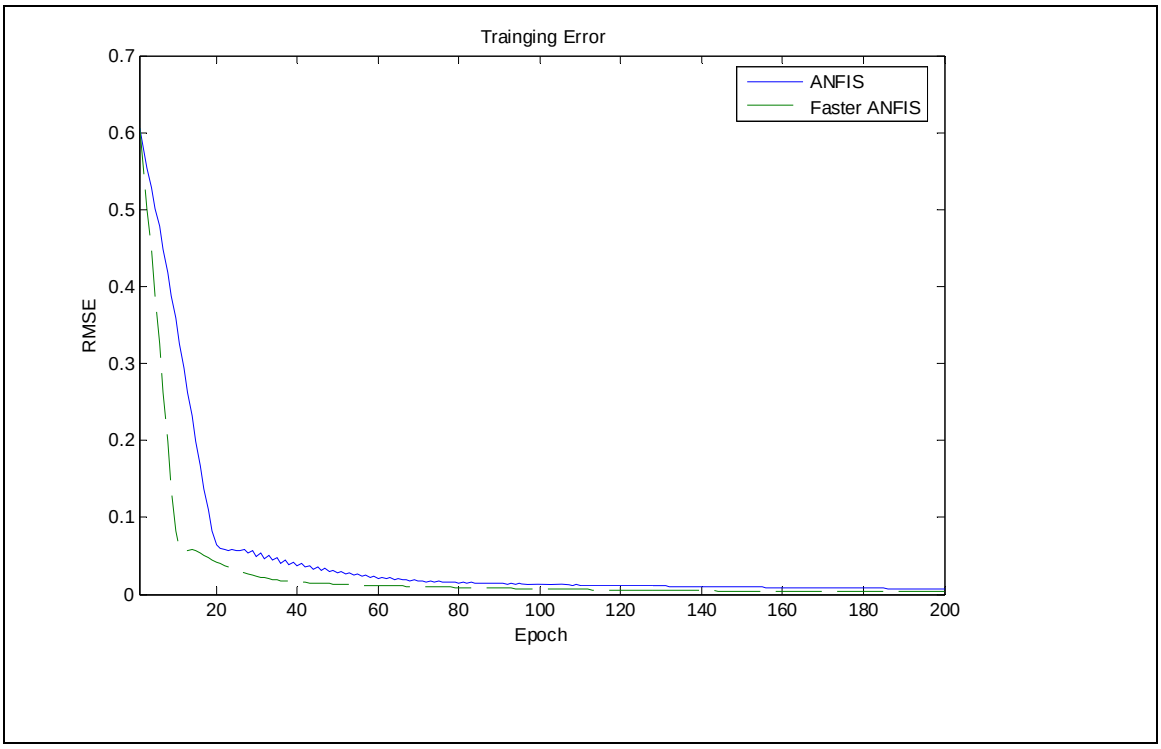


Figure 7.6 (a) Training RMSE for ANFIS versus FANFIS.

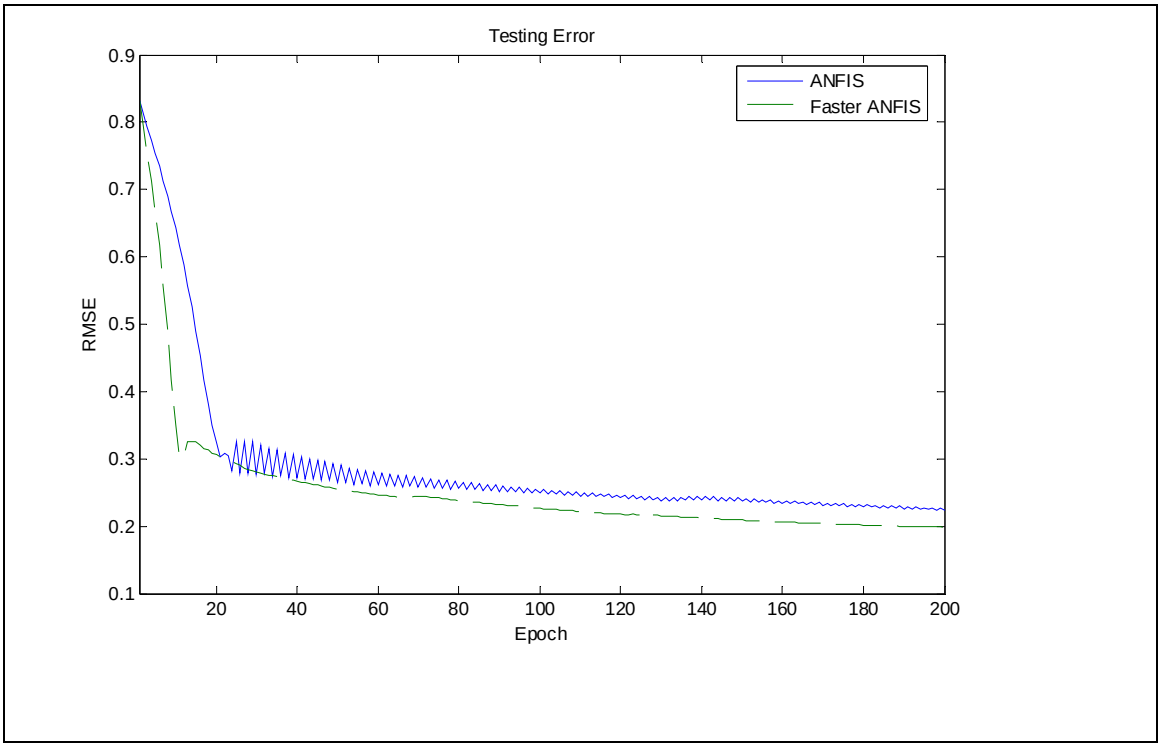


Figure 7.6 (b) Testing RMSE for ANFIS versus FANFIS.

Figures 7.7 and 7.8 show the predicted output of the testing dataset for FANFIS and ANFIS. In fine scale (Figure 7.9) we can see that the FANFIS is closer to the actual output than ANFIS.

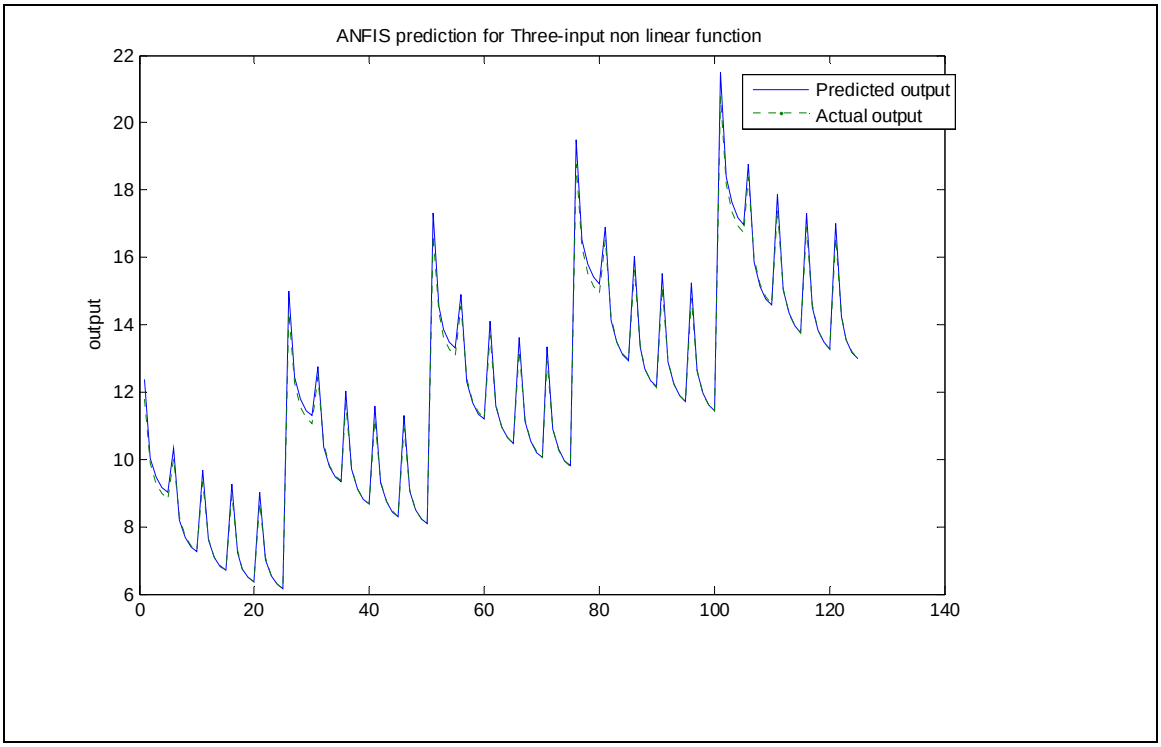


Figure 7.7 ANFIS prediction for three-input non linear function.

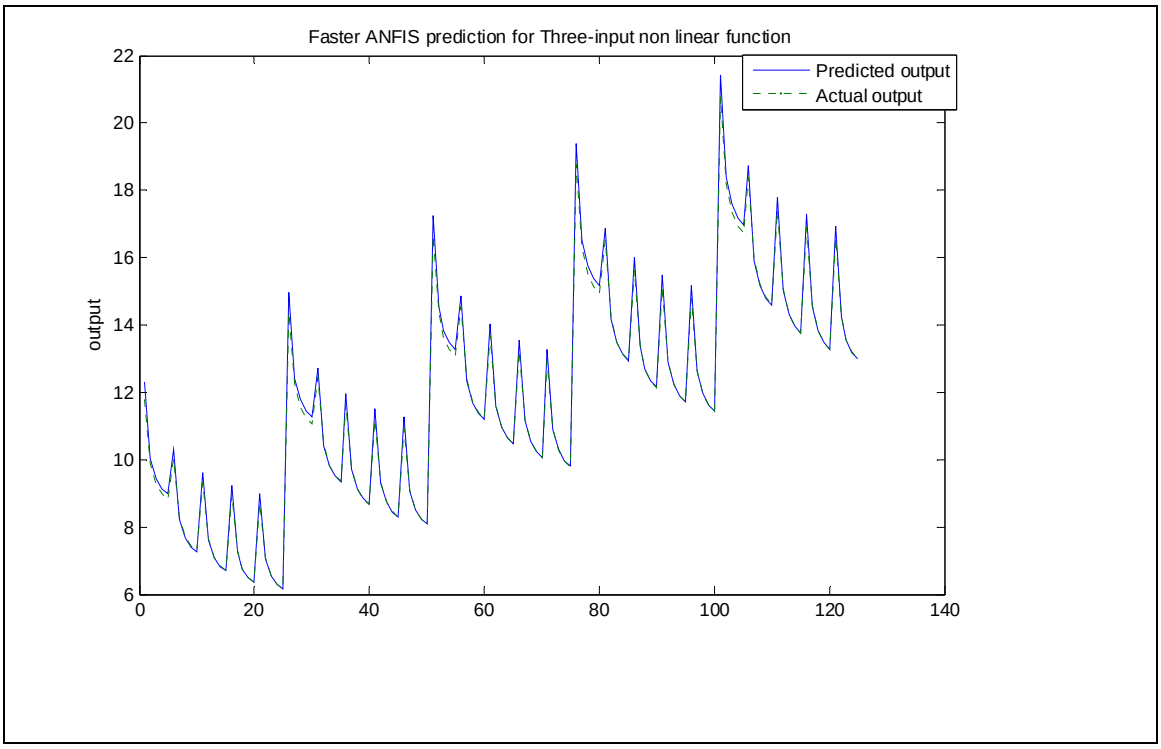


Figure 7.8 FANFIS prediction for three-input non linear function.

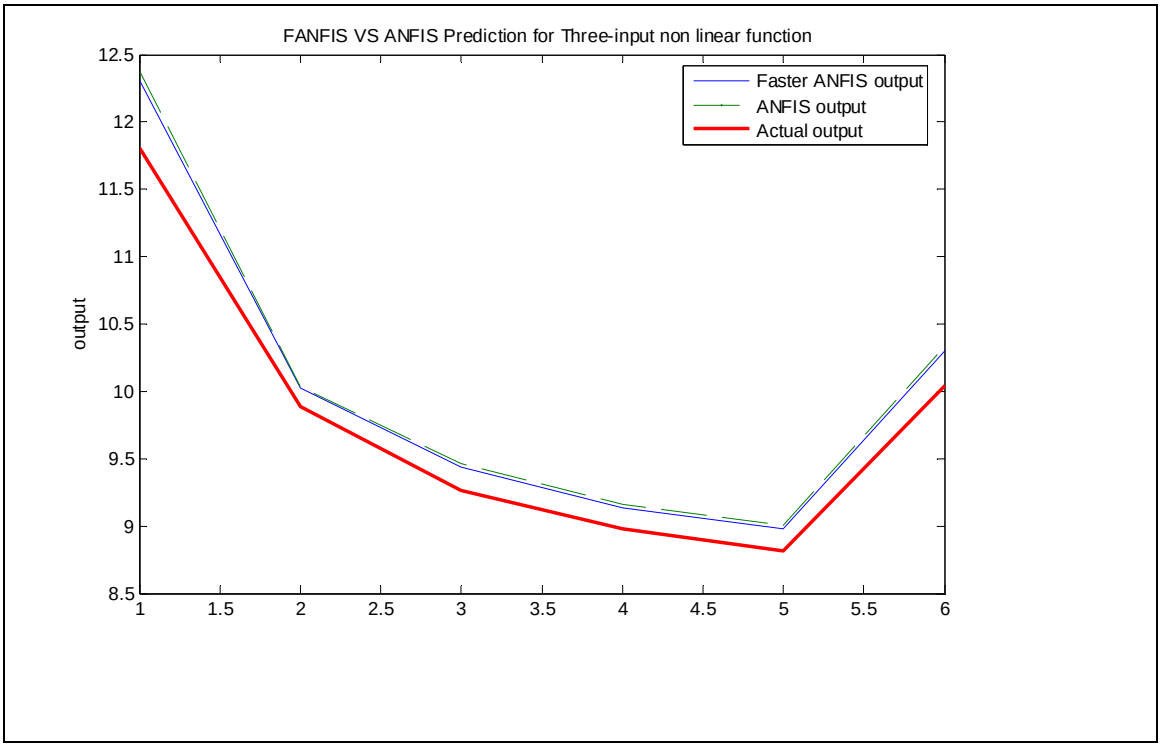


Figure 7.9 FANFIS versus ANFIS predicted output in finer scale.

To allow comparison with the results in [37, 38] we shall use the same performance index which is

$$APE = average\ percentage\ error = \frac{1}{P}\sum_{i=1}^{P}\left|\frac{T(i)-O(i)}{T(i)}\right| \times 100\%$$

where P is the number of data pairs, $T(i)$ and $O(i)$ are the i^{th} desired output and predicted output, respectively.

After 200 epochs the final results were $APE_{Training} = 0.0195\%$ and $APE_{Testing} = 0.9197\%$ for FANFIS, which are listed in Table 7.2 along with the results of ANFIS and other earlier work [37, 38].

Model	Training Error	Testing Error	Time in second
FANFIS	0.0195%	0.9197%	3.6
ANFIS	0.0389%	1.0562%	7.9
GMDH model [37]	4.7%	5.7%	N/A
Fuzzy model 1 [38]	1.5%	2.1%	N/A
Fuzzy model 2 [38]	0.59%	3.4%	N/A

Table 7.2 Comparison of average percentage errors with different models

7.3.3 Identification of Dynamical Systems

Here we use FANFIS to identify the dynamical system in [39]. In that paper a 1-20-10-1 back-propagation multilayer perceptron was used to identify a nonlinear component in a dynamical system. The plant under consideration is governed by the following difference equation:

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f\left(x(k)\right),$$

where $y(k)$ and $x(k) = \sin\left(\frac{2\pi k}{250}\right)$ are the output and input, respectively, at time step *k*. The unknown function $f(\cdot)$ has the form

$$f(x) = 0.6\sin(\pi x) + 0.3\sin(3\pi x) + 0.1\sin(5\pi x).$$

In order to identify the plant, a series-parallel model governed by the difference equation

$$\hat{y}(k+1) = 0.3\hat{y}(k) + 0.6\hat{y}(k-1) + F\left(x(k)\right)$$

was used. The multilayer perceptron in [39] failed to follow the plant when the adaptation was stopped at $k = 500$, and the identification procedure had to continue for 50,000 time steps using a random input.

For the FANFIS simulation we used a training data set of size 101, sampled uniformly from the input range [-1, 1] to identify $F(\cdot)$ in the difference equation and since it is a system identification problem using testing data set is not necessary. Figures 7.10, 7.11, 7.12 show the results for FANFIS after 50 epochs of learning when the number of MFs is 3, 5 and 7 respectively.

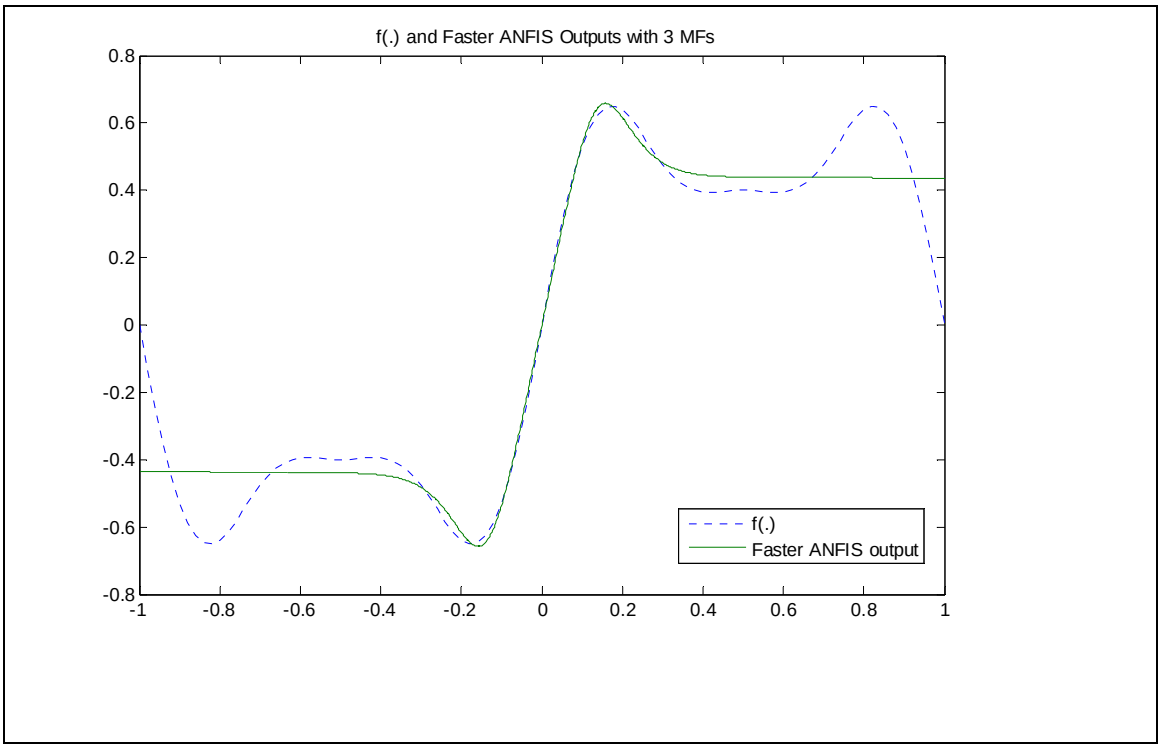


Figure 7.10 FANFIS identification of *F(.)* using 3 membership functions

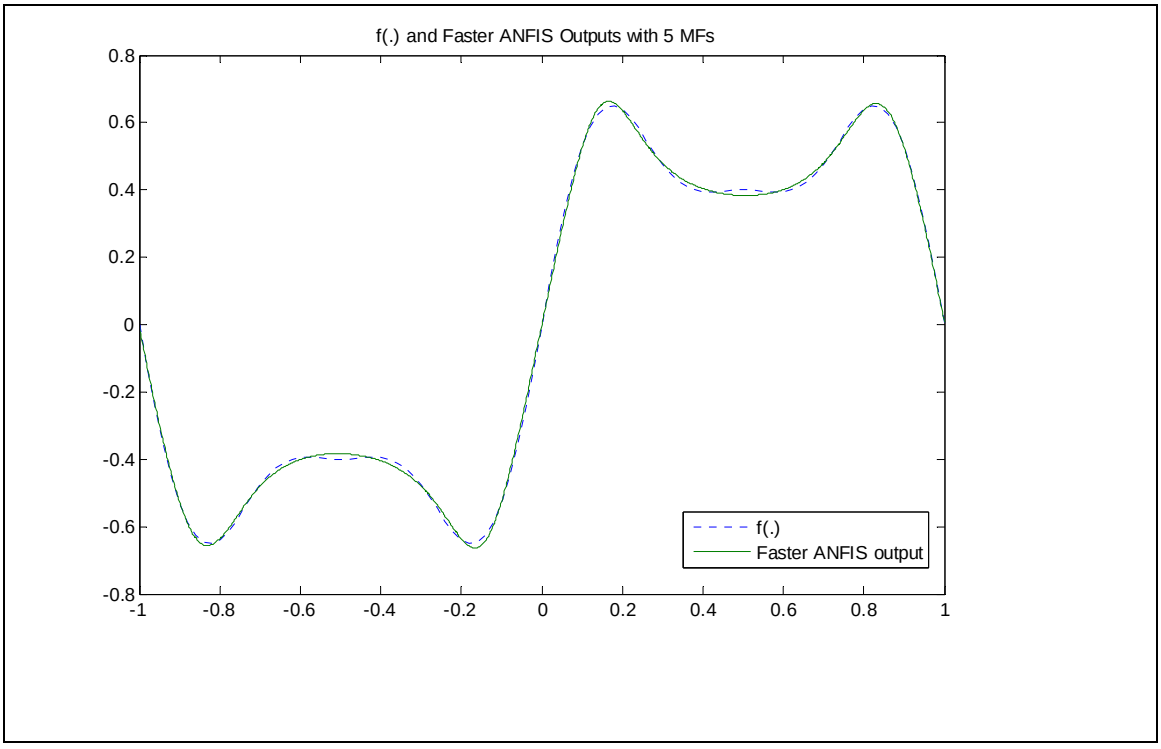


Figure 7.11 FANFIS identification of F(.) using 5 membership functions

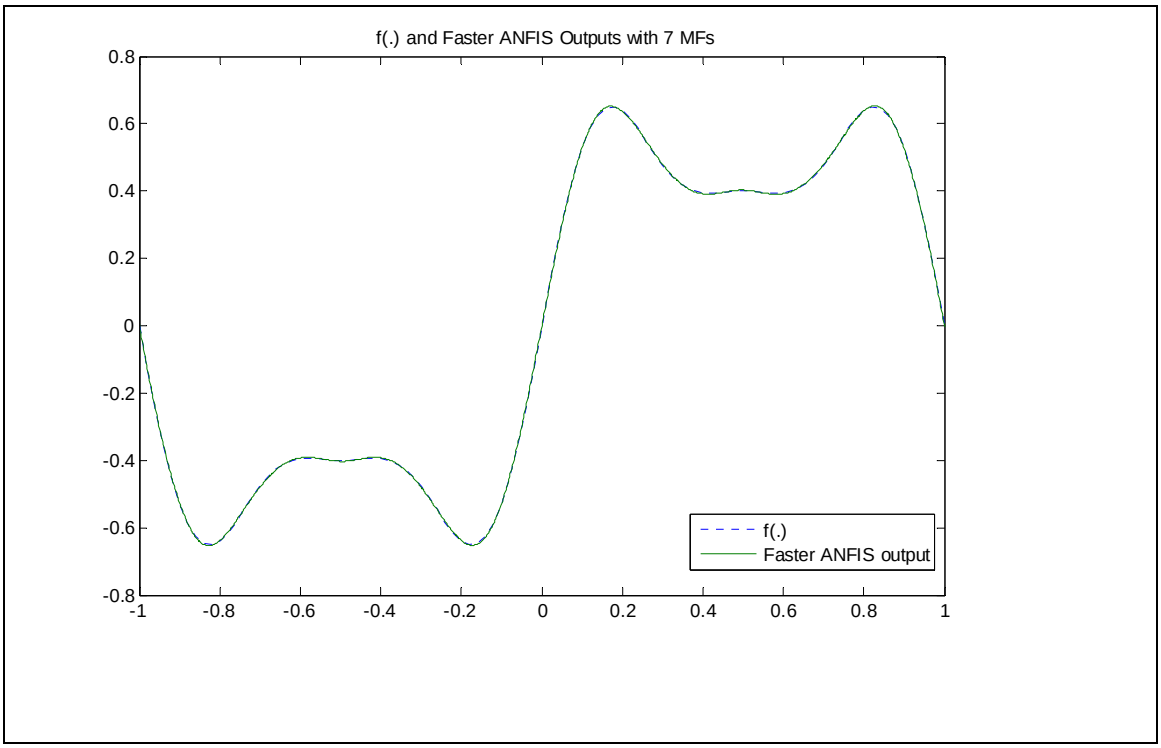


Figure 7.11 FANFIS identification of *F(.)* using 7 membership functions

The results show that FANFIS can identify the function $F(\cdot)$ very well and the performance increases when the number of membership function increases. Here FANFIS achieves the best performance at the cost of computational time which are 0.141, 0.172 and 0.375, seconds for 3, 5 and 7 membership functions respectively.

In comparison FANFIS outperformed ANFIS in every case where the computational time for ANFIS are 0.344, 0.453 and 0.563, seconds for 3, 5 and 7 membership functions respectively. Figure 7.12, 7.13 and 7.14, show the comparison between the training error of FANFIS and ANFIS, where FANFIS has a faster and smoother convergence in every case.

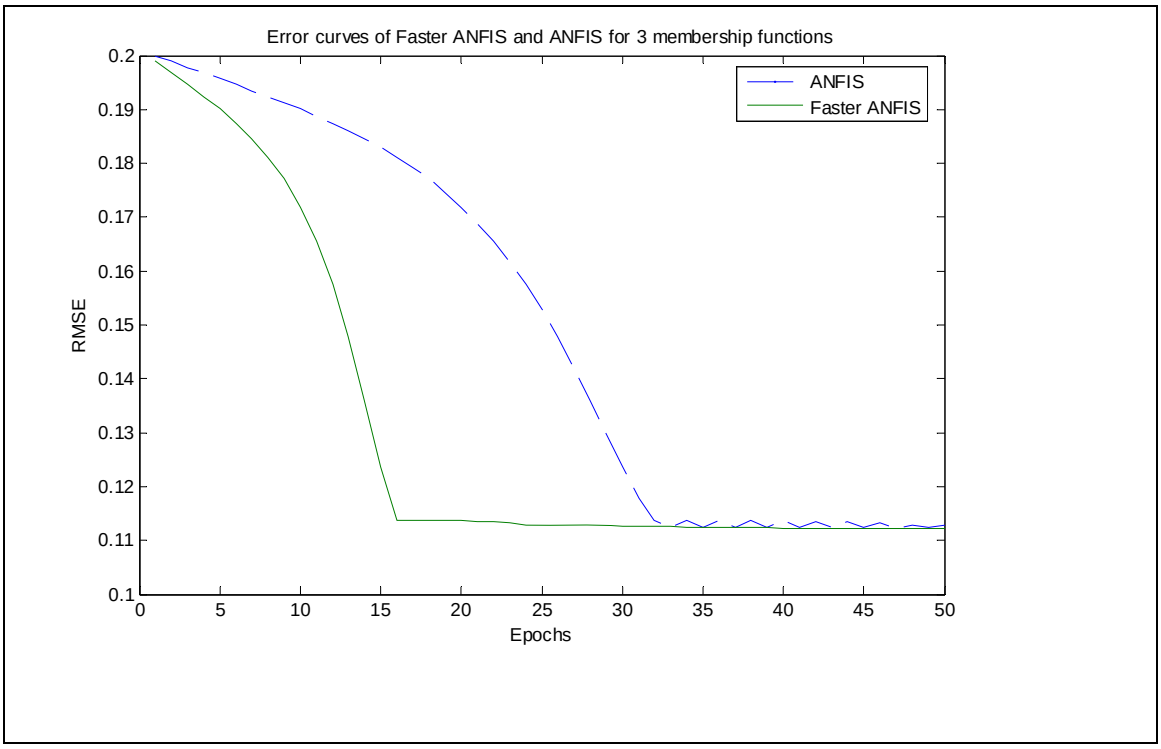


Figure 7.12 FANFIS RMSE versus ANFIS RMSE for 3 membership functions

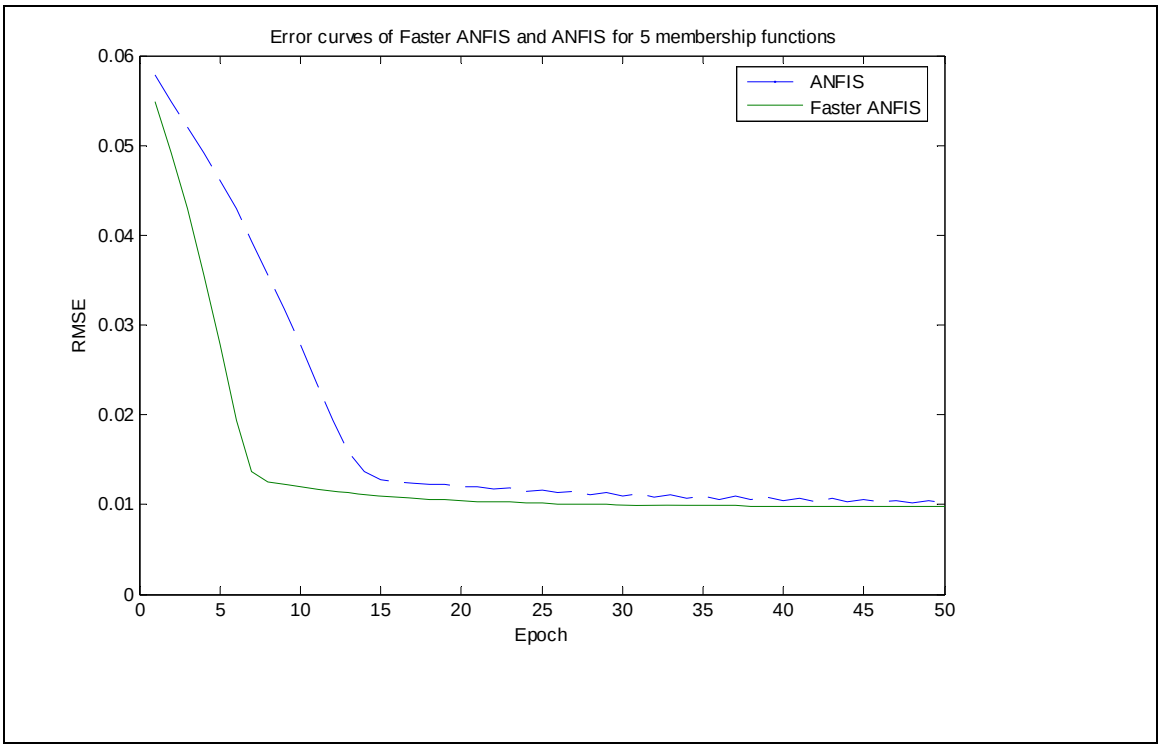


Figure 7.13 FANFIS RMSE versus ANFIS RMSE for 5 membership functions

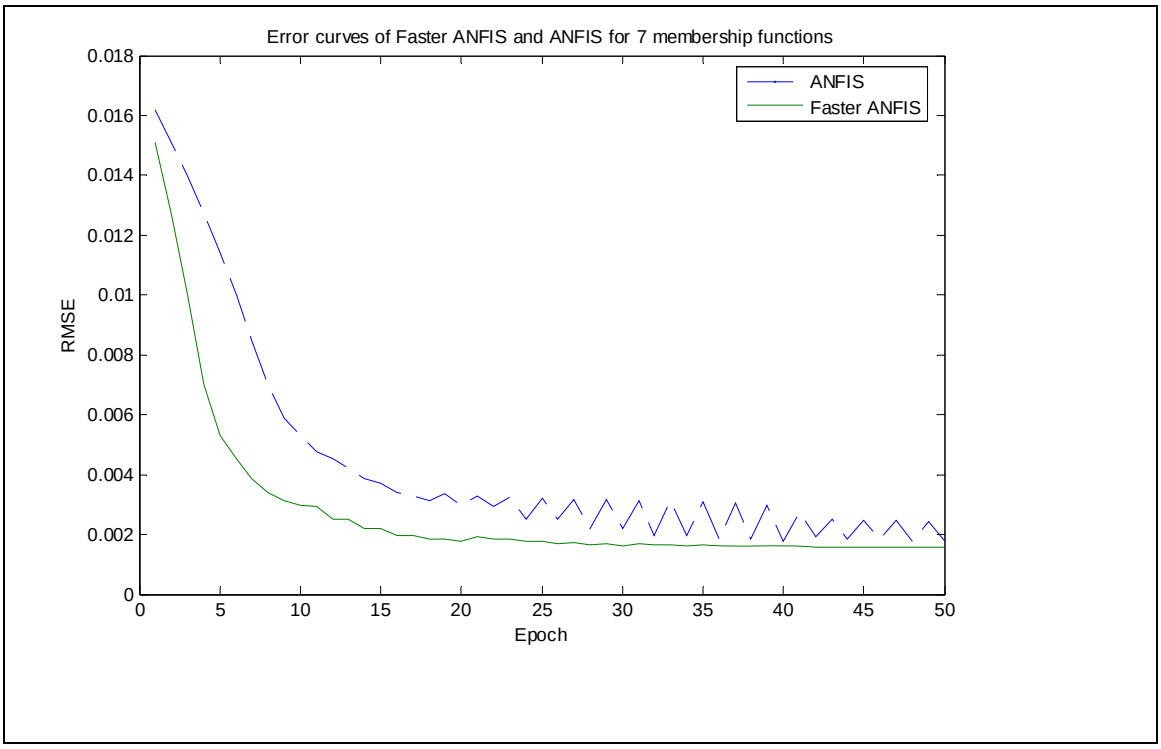


Figure 7.14 FANFIS RMSE versus ANFIS RMSE for 7 membership functions

Table 7.3 summarizes the results for FANFIS and ANFIS in this identification of dynamical system problem.

Model	FANFIS		ANFIS	
	RMSE	Time in Second	RMSE	Time in Second
7 MFs	0.0016	0.375	0.0018	0.563
5 MFs	0.0097	0.172	0.0101	0.453
3 MFs	0.1118	0.141	0.1120	0.344

Table 7.3 Summarized results for identification of dynamical system

Note that all the models in Table 7.3 take only 50 learning steps to identify the system, while the original multilayer perceptron model in [39] failed to identify the system after 500 learning steps. The error of the original model was not reported.

However, the results do not imply that more membership functions are better for the system. There are a trade-off between bias and variance [48] where a classifier partitions the input space into regions. When these regions are too large, the degree of fit to an accurate partitioning of the instance space will be poor, increasing error rates. This effect is called *bias*. When the regions are too small, the probability that individual regions are labelled with the wrong class is increased. This effect, called *variance*, also increases error rates. A way to solve this problem is to regulate the error goal and either fixes the number of epoch of training and increase the number of membership function until the goal is met.

7.3.4 Forecasting Stock market price

In this example we use FANFIS to forecast stock market prices as a comparison to my previous research [40] using the Martingale hypothesis model [44, 45] which is the most common hypothesis about stock prices (This hypothesis can be describe in economist terms as the Efficient Market Hypothesis [47]), Box and Jenkins models [41], Bayesian Dynamic linear model [42] and Fuzzy neural network by Hobbs and Bourbakis [43] were used to forecast New Zealand stock market daily closing prices.

Here we apply the methods of the same data set namely Air New Zealand Ltd. “A”, Brierley Investment Ltd., Carter Holt Harvey Ltd., Lion Nathan Ltd. and Telecom Corporation of New Zealand Ltd. The data span 1872 trading days, with the first observation being 1st of May 1992 and the last observation being 7th of October 1999.

Also we employ the same trading strategy as in my previous research. This trading strategy is based on the assumption that if tomorrow’s closing price is predicted to be less than today’s closing price we will sell at tomorrow’s closing price because the stock is predicted to have reached the local maximum value. On other hand if tomorrow’s closing price is more than today’s closing price we will buy at tomorrow’s closing price because the stock is predicted to have reached the local minimum value.

This trading strategy was chosen in my previous work because the data available were stock closing prices, available at the end of the trading day. Therefore by predicting tomorrow's closing prices we can only buy, sell or hold the stock at closing tomorrow. The strategy is only a simple one because it was not the intention of my previous study to research into trading strategies, which can be very complex and irreverent for current purposes.

The trading strategy simulation is summarized as follows.

1. Start with an initial amount (cash balance) of \$1000 let n be the number of observations and $i = 1$.
2. Analyze each stock from *i* to *n* and predict the price for the $(n+i)^{\text{th}}$ observation (using FANFIS in this case).
3. If the predicted price in step 2 is more than the price for n^{th} observation and the cash balance is more than 0, buy the stock at the $(n+i)^{\text{th}}$ observation price using all cash available.
4. If the predicted price in step 2 is less than price for n^{th} observation and the number of stocks held is more than 0, sell all of the stock at the $(n+1)^{\text{th}}$ observation price.
5. If the situation does not satisfy conditions 3 or 4 do not trade.
6. Increase n and *i* by 1 and go to step 2.

The FANFIS training data set consists of 500 input output data pairs for each j^{th} predictions $(j = 1,...,1368)$ which are in the form of

$$\left[s(t-4),\ s(t-3),\ s(t-2),\ s(t-1); s(t)\right],$$

where $s(t)$ is a closing stock price at time $t = 4 + j,...,504 + j$.

Two membership functions were used for each input. The rest of the data set is for testing the model and since the training set changes for each prediction, the plots of the membership function parameters for each prediction are shown in Appendix II.

Figures 7.15 to 7.24 show FANFIS predictions versus the actual stock closing prices and their residuals. There are some anomalies in predictions on Carter Holt Harvey and Lion Nathan's closing prices. These are caused by the sudden extreme price drop on both stocks. In general there are evidences of positive autocorrelation in the prediction residuals, usually after small shocks in prices. These findings are not surprising since the system need to re-adjust itself to cope with shocks and possible market corrections in stock prices.

However, the FANFIS models were able to adapt to these conditions and the predictions return to their usual accuracy after a short period of time. Overall the FANFIS predictions are very close to the actual stock closing prices.

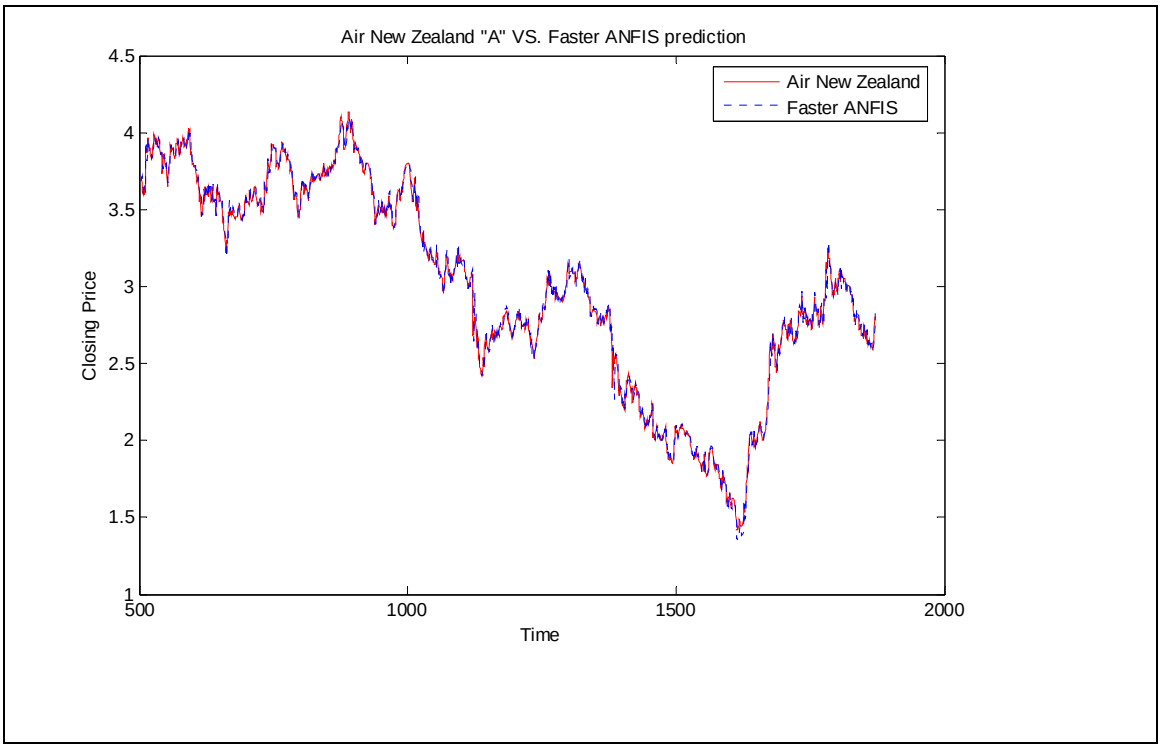


Figure 7.15 FANFIS prediction of Air New Zealand Ltd. "A" closing price.

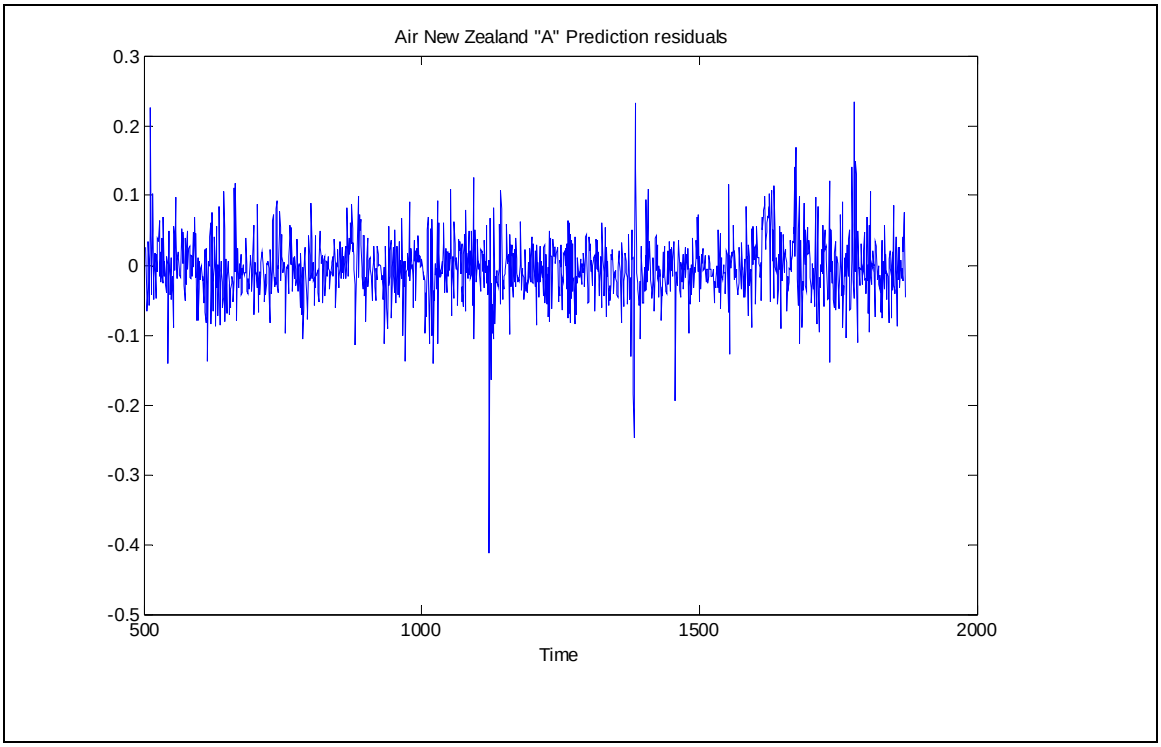


Figure 7.16 FANFIS prediction residuals of Air New Zealand Ltd. "A"

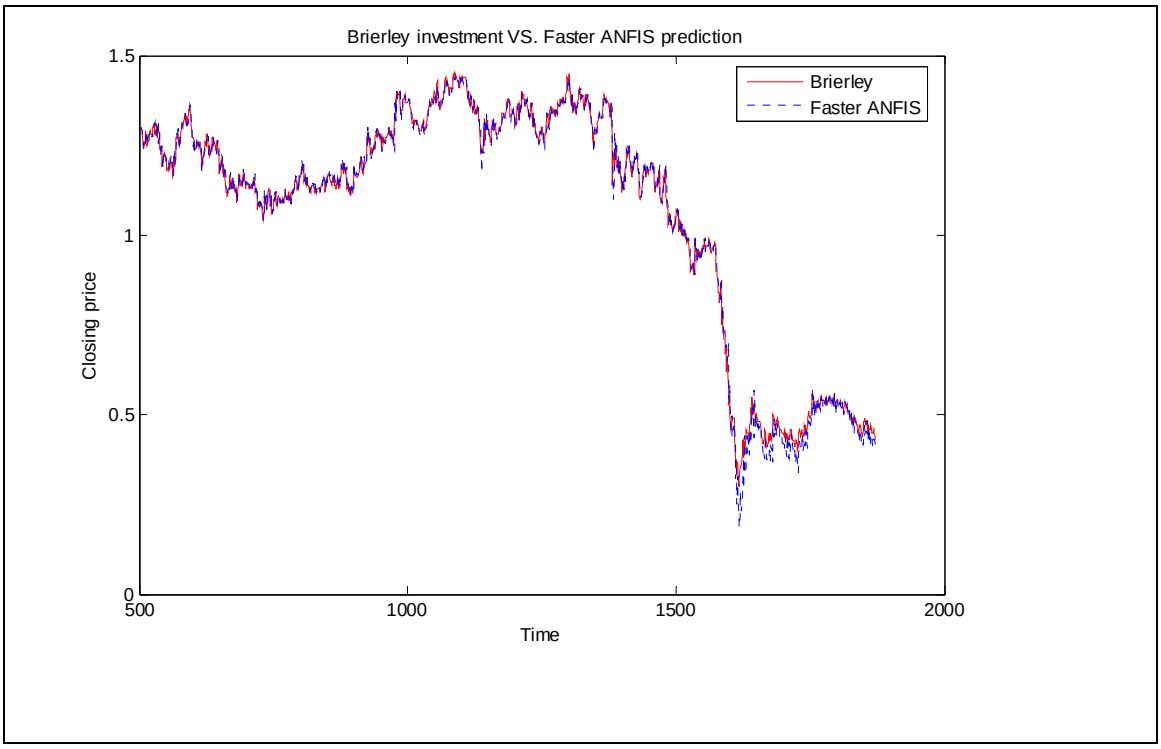


Figure 7.17 FANFIS prediction of Brierley Investment Ltd. closing price.

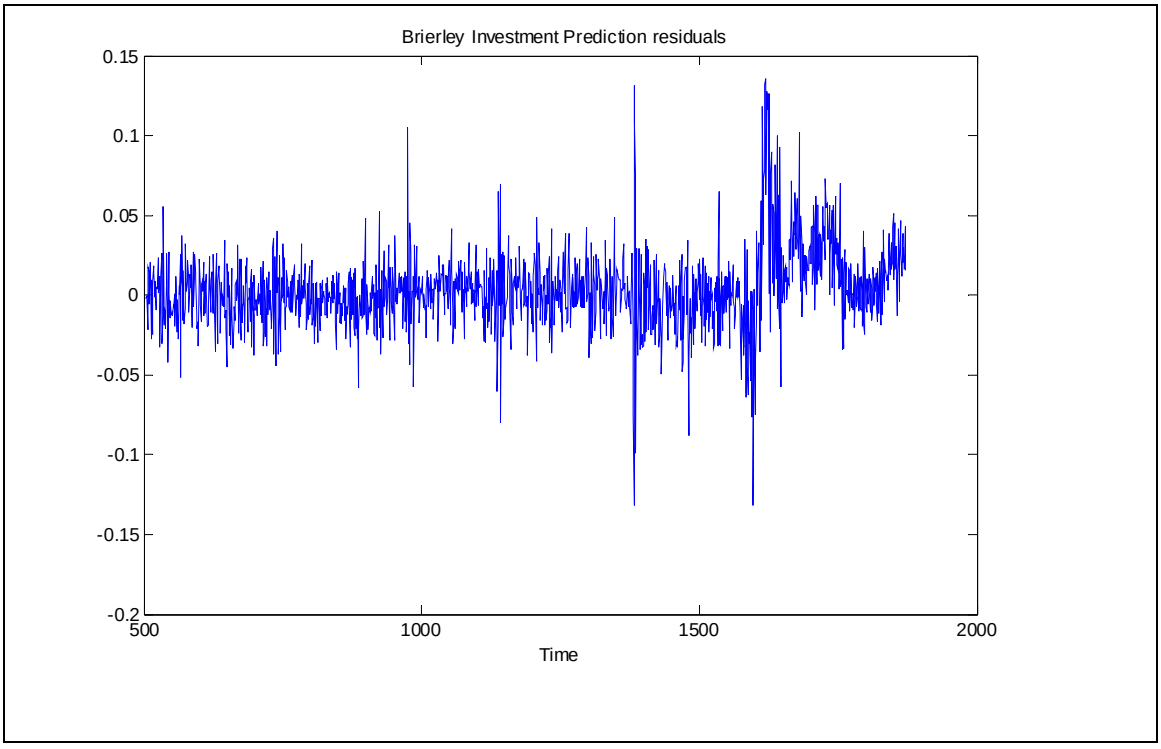


Figure 7.18 FANFIS prediction residuals of Brierley Investment Ltd.

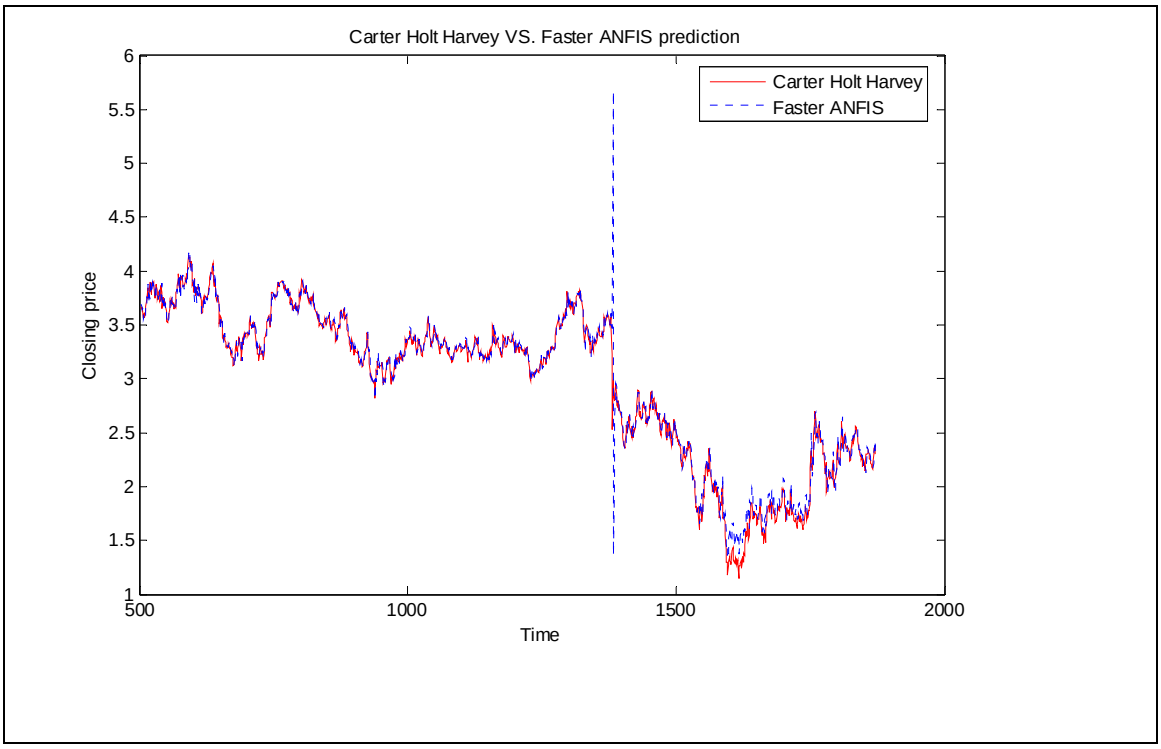


Figure 7.19 FANFIS prediction of Carter Holt Harvey Ltd. Closing price.

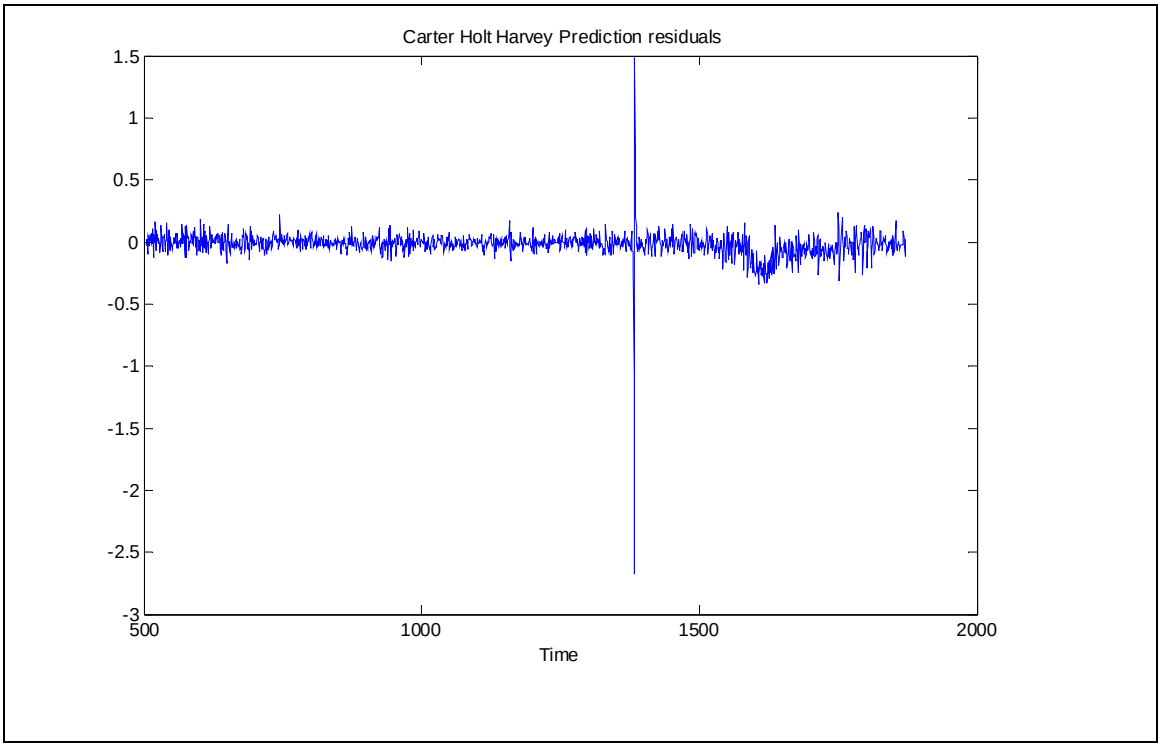


Figure 7.20 FANFIS prediction residuals of Carter Holt Harvey Ltd.

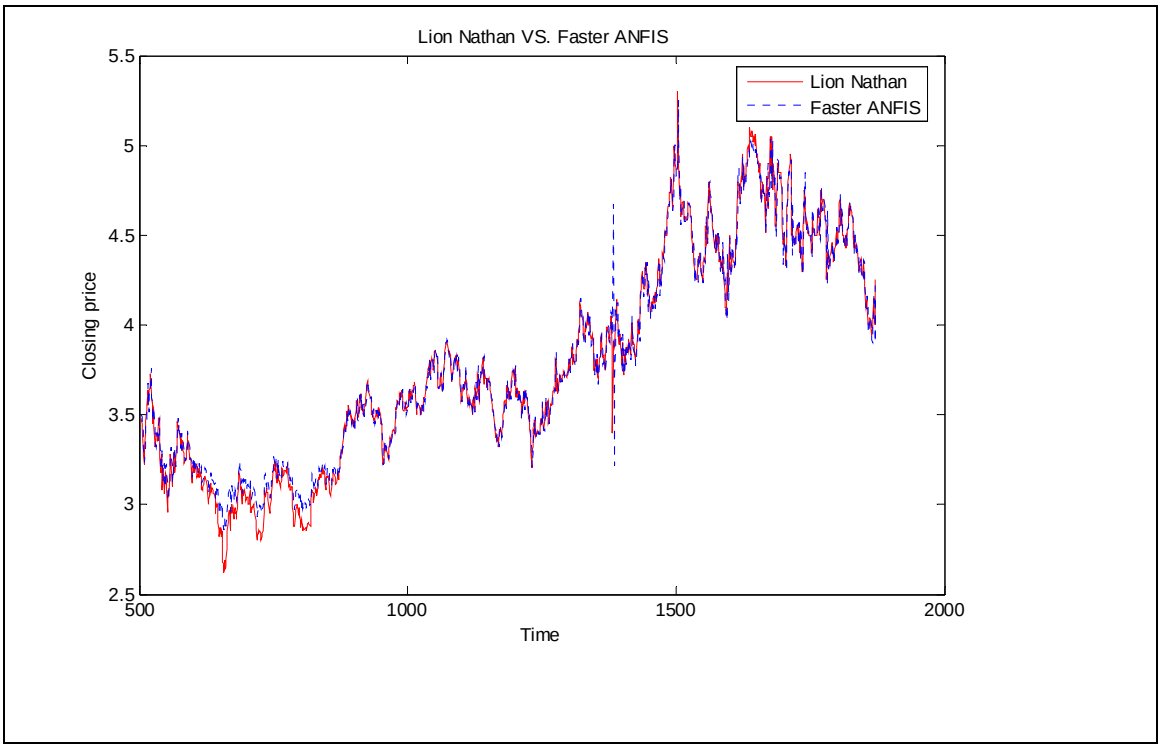


Figure 7.21 FANFIS prediction of Lion Nathan Ltd. Closing price

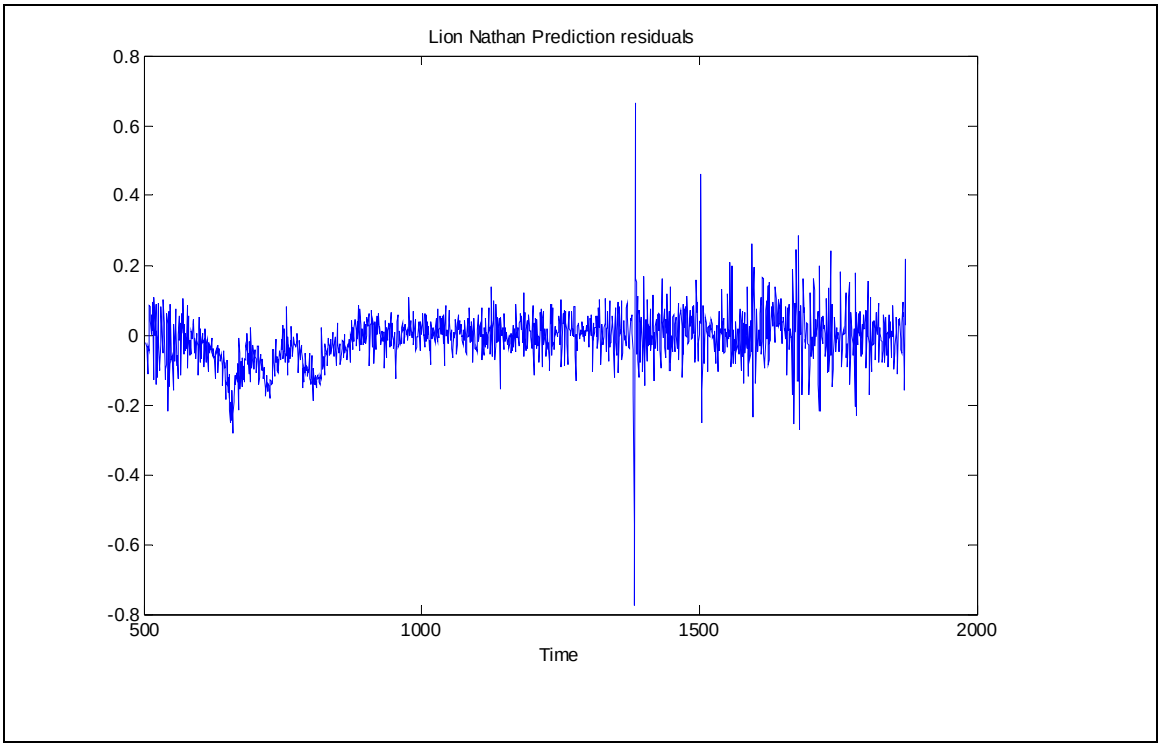


Figure 7.22 FANFIS prediction residuals of Lion Nathan Ltd.

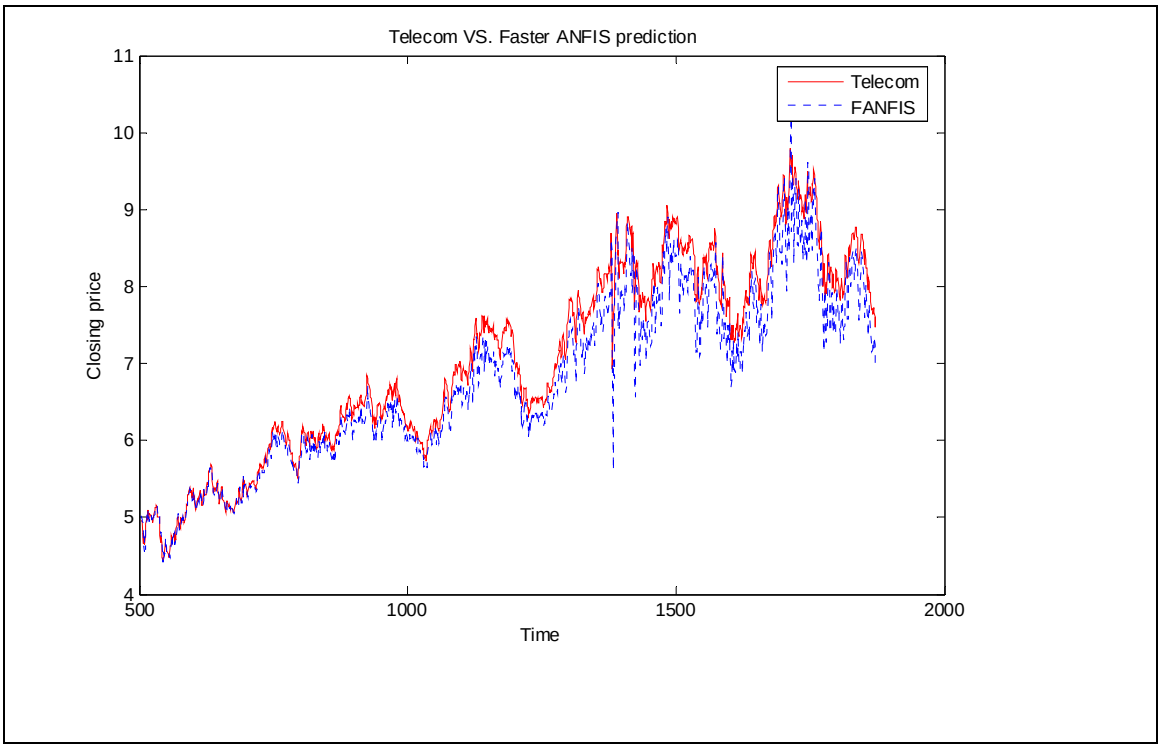


Figure 7.23 FANFIS prediction of Telecom Corporation Ltd. Closing price

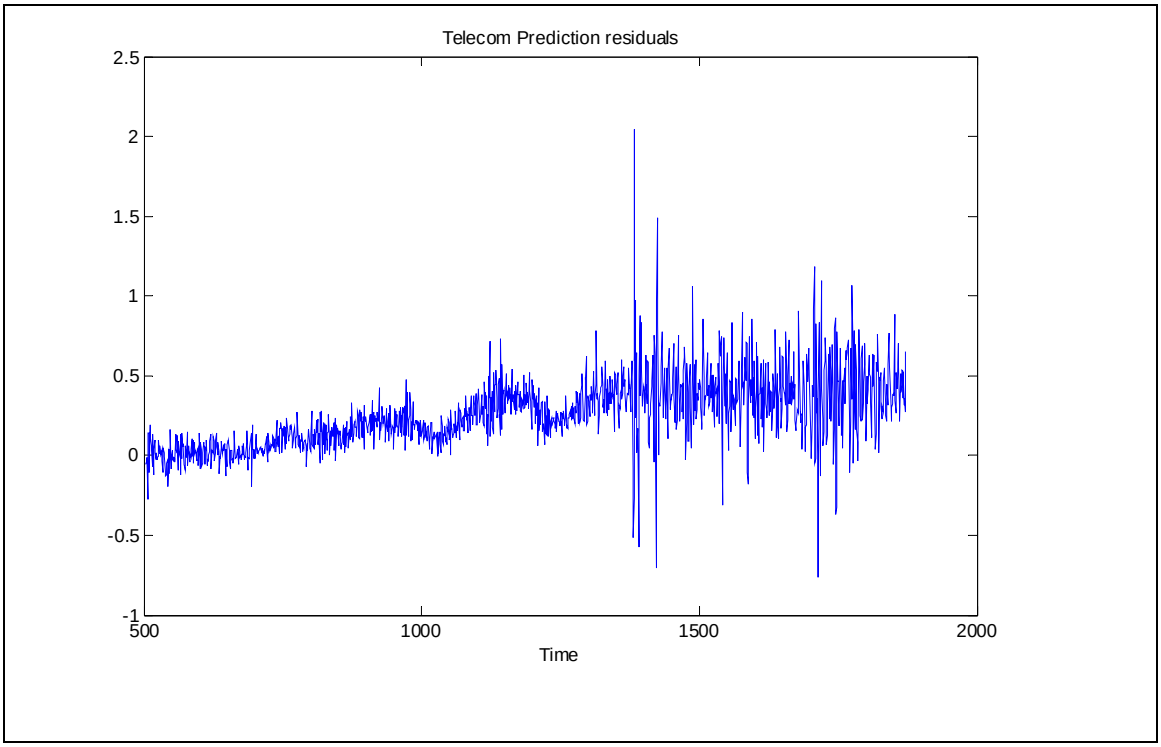


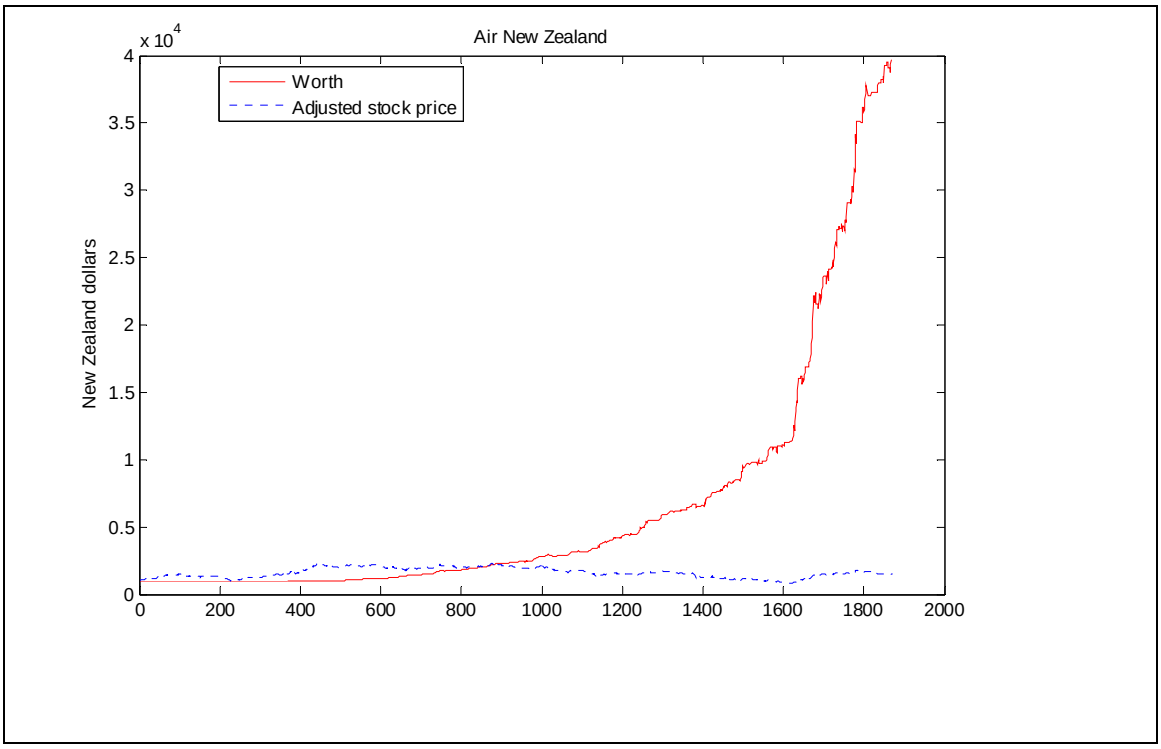
Figure 7.24 FANFIS prediction residuals of Telecom Corporation Ltd.

To be able to compare the results with other models in my previous study, the following Tables 7.4 to 7.8 (the last 4 rows of each table are the results of my previous research [40]) and Figures 7.25 to 7.29 are the results of using FANFIS with the trading strategy mentioned in the beginning of this section.

Note that the results are for either cash balance or the amount of stocks held in dollars noted here as "worth" and in Figures 7.25 to 7.29 worth are compared with the adjusted stock prices which are based on the martingale hypothesis of the stock's closing price. The results show that the FANFIS outperformed all other models using the same trading strategy.

Overall the FANFIS produced worth averagely forty-six times more than the Martingale hypothesis model, forty times more than Box and Jenkins models, forty-four times more than Bayesian Dynamic linear model and twenty times more than Hobbs and Bourbakis's Fuzzy neural network.

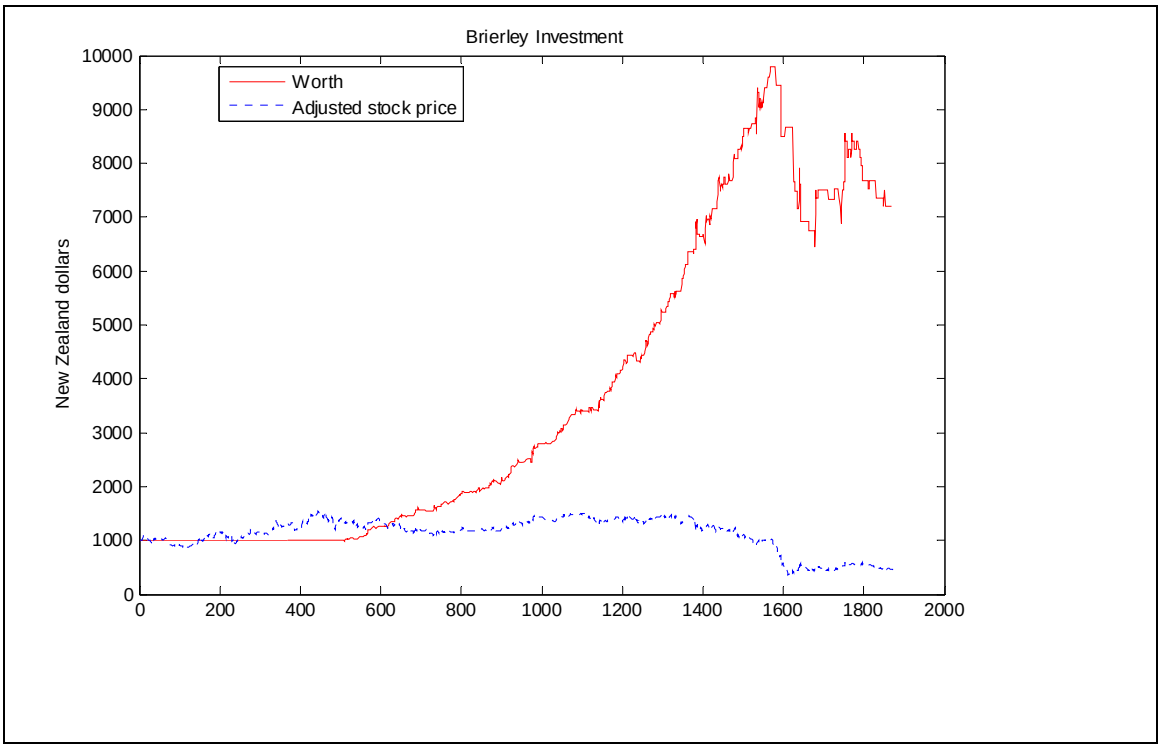
While the considerations of transaction cost have been ignored in this research, with the technology of online trading the traders now can trade stocks via the internet with much less transaction cost required so that the cost of trading will not considerably affect the conclusions.



7.25 Trading simulation result for Air New Zealand Ltd. "A".

Models	Mean	Standard Deviation	Maximum	Minimum	Final Cash Balance
FANFIS	6454.33	9258.04	39675.91	989.01	39675.91
Martingale	1606.09	370.99	2281.77	784.53	1535.91
Box and Jenkins	1489.25	297.72	2201.66	985.51	1912.70
Bayesian DLM	1254.40	148.24	1516.47	941.34	1155.46
Fuzzy Neural	2376.75	797.43	3858.27	1000.00	2996.06

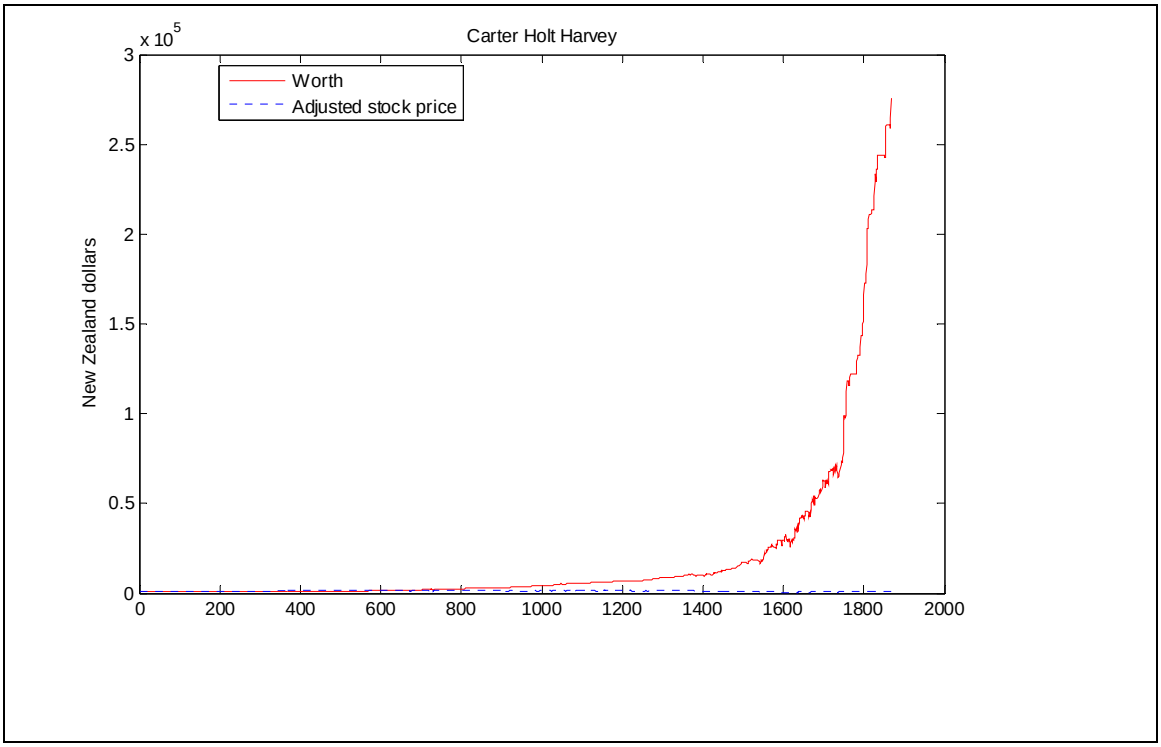
Table 7.4 Comparative results for Air New Zealand's stock closing price.



7.26 Trading simulation result for Brierley Investment Ltd.

Models	Mean	Standard Deviation	Maximum	Minimum	Final Cash Balance
FANFIS	3640.93	2836.37	9804.84	992.06	7200.04
Martingale	1122.44	303.32	1541.66	312.50	447.91
Box and Jenkins	1156.54	196.24	1541.49	560.33	1131.04
Bayesian DLM	1166.02	315.02	1727.63	477.56	507.30
Fuzzy Neural	1135.21	180.37	1504.98	631.85	695.81

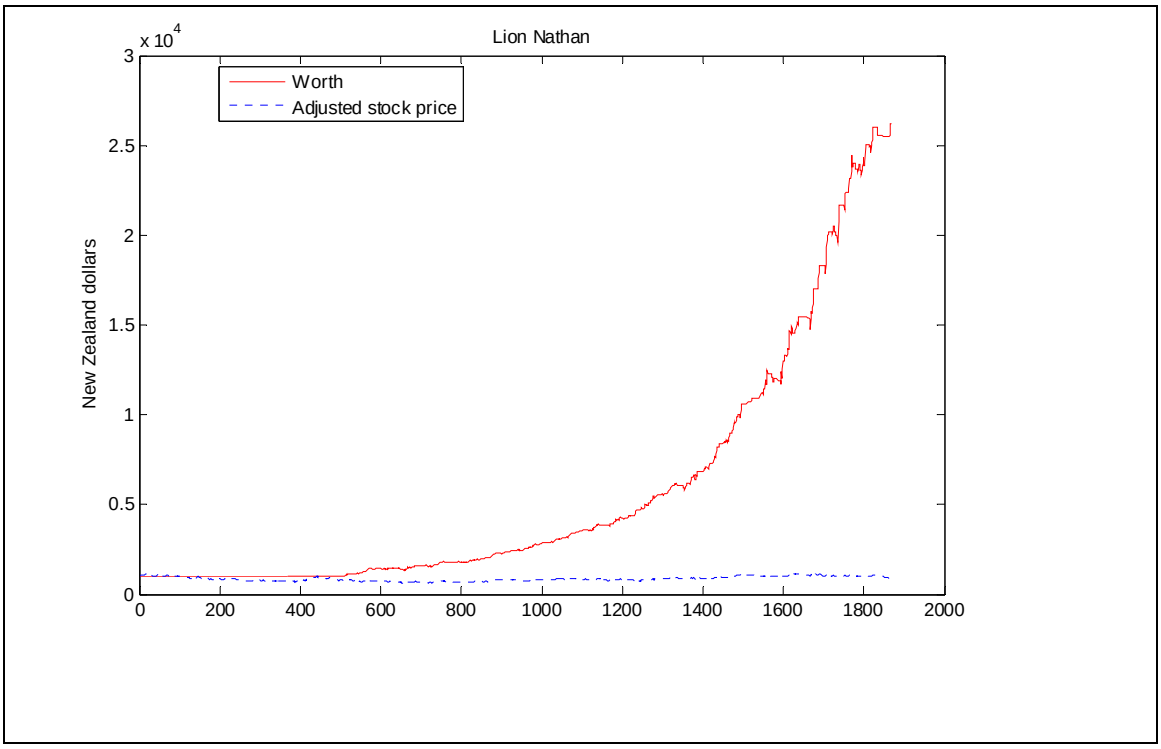
Table 7.5 Comparative results for Brierley Investment's stock closing price.



7.27 Trading simulation result for Carter Holt Harvey Ltd.

Models	Mean	Standard Deviation	Maximum	Minimum	Final Cash Balance
FANFIS	20064.39	47480.18	275403.02	991.66	275403.02
Martingale	1199.51	263.89	1663.99	459.99	940.00
Box and Jenkins	1414.02	257.51	1964.03	741.11	1899.76
Bayesian DLM	1352.99	257.46	1886.45	759.41	1152.1
Fuzzy Neural	1663.12	325.69	2487.60	949.80	2403.99

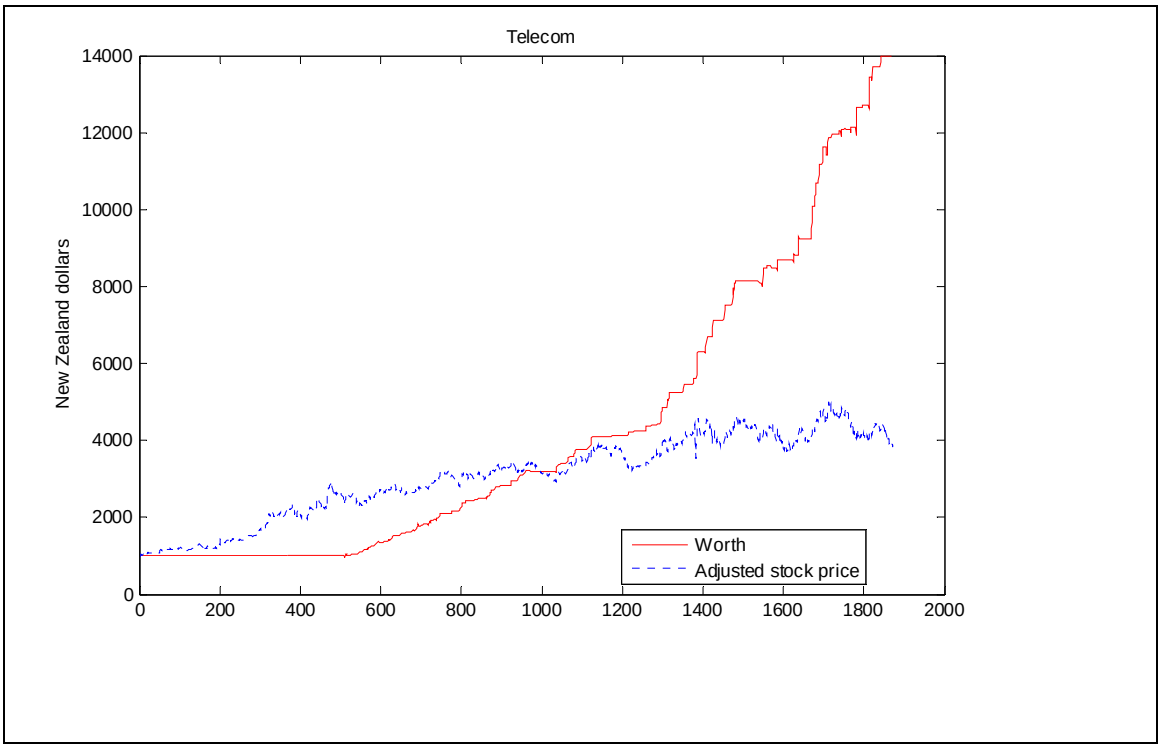
Table 7.6 Comparative results for Carter Holt Harvey's stock closing price.



7.28 Trading simulation result for Lion Nathan Ltd.

Models	Mean	Standard Deviation	Maximum	Minimum	Final Cash Balance
FANFIS	5703.44	6820.89	26181.89	981.70	26181.89
Martingale	867.67	128.81	1210.04	598.17	970.31
Box and Jenkins	1254.40	148.24	1516.47	941.34	1155.46
Bayesian DLM	1049.08	69.73	1293.49	874.46	1077.41
Fuzzy Neural	2090.02	845.11	3928.87	1000.00	3834.20

Table 7.7 Comparative results for Lion Nathan's stock closing price.



7.29 Trading simulation result for Telecom Corporation of New Zealand Ltd.

Models	Mean	Standard Deviation	Maximum	Minimum	Final Cash Balance
FANFIS	4230.76	3715.78	13996.56	966.66	13996.56
Martingale	3065.42	1059.53	5000.00	994.89	3882.65
Box and Jenkins	2376.75	797.43	3858.27	1000.00	2996.06
Bayesian DLM	3080.40	1135.37	5506.86	1000.00	4290.89
Fuzzy Neural	3764.85	1942.83	8034.43	989.95	7506.39

Table 7.8 Comparative results for Telecom's stock closing price.

Chapter 8

8 Conclusion

This thesis summarized all the ideas necessary to develop the Faster Adaptive Network based Fuzzy Inference System (FANFIS) such as fuzzy logic, fuzzy sets, membership functions, fuzzy rules, fuzzy reasoning, fuzzy inference systems, adaptive networks, back-propagation algorithm, hybrid learning rules and adaptive network based fuzzy inference system (ANFIS). We discussed several shortcomings in the learning procedure of ANFIS such as the ill-conditioned matrix in the least square estimate, the Stein's phenomenon and the time consuming algorithm of the back-propagation learning procedure.

Therefore we have developed FANFIS in this thesis to deal with the aforementioned shortcomings. FANFIS is truly "faster" than ANFIS because the system does not uses the least square estimator or the gradient descents back-propagation in its learning procedure which has several shortcomings in ANFIS. FANFIS used the same learning procedure for both premise (linear) and consequence (non-linear) parameters. This learning procedure only required the *direction* of the change of the parameters which is much simpler to calculate than the *size* of the change.

We tested FANFIS against ANFIS and other competing models in different applications such as chaotic time series predictions, dynamical system identification, modelling non-linear function. The results show that FANFIS out-performs ANFIS and the other competing models in speed and accuracy.

Also we used FANFIS to predict five major stock prices in New Zealand. The results have shown that FANFIS is able to estimate the stock closing prices accurately with only few anomalies in the predictions when there are extreme changes stock prices. Using the

FANFIS predictions, stocks trading was simulated by employing a simple trading strategy against other models such as the Martingale hypothesis model which is the most common hypothesis about stock prices, Box and Jenkins time series models which are the most popular time series models, Bayesian Dynamic linear models and the Neurofuzzy simulator for stock investing modeled by Hobbs and Bourbakis. The results have shown that FANFIS is the most profitable model.

I believed that further research can be done in using FANFIS in such areas as

- applications that require multiple outputs,
- forecasting nonlinear econometrics models,
- finding the nonlinear relationship between different time series.

FANFIS can be developed for such application by

- finding an efficient way to choose the number of membership functions,
- finding new adaptable membership functions,
- improving the learning procedures.

Appendix I

Initial and final membership functions in Sections 7.3.1 to 7.3.4

Note that all the membership functions listed here are the generalized bell membership function. This is because the generalized bell membership function is the most adaptable membership function in fuzzy sets literature as it is shown in section 2.3 of this thesis. The initial membership functions are selected by grid partitioning. Here is the method of selecting the initial generalized bell membership functions by grid partitioning for each FANFIS input:

Recall the formula for generalized bell membership functions in definition 2.5,

*A **generalized bell membership function** is specified by three parameters $\{a,b,c\}$:*

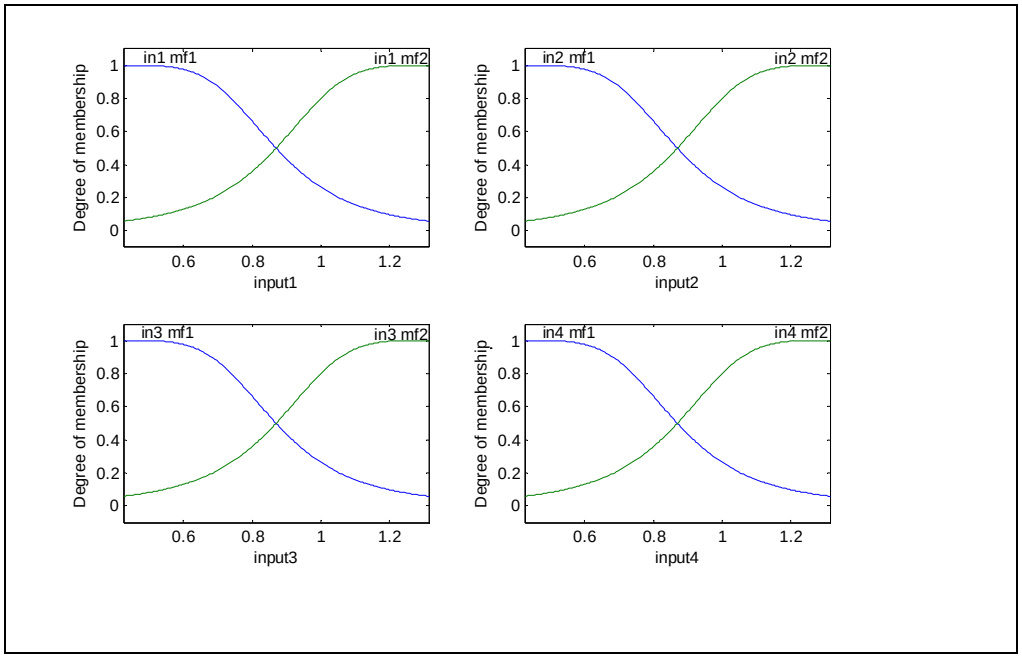
$$bell(x;a,b,c)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}.$$

Let

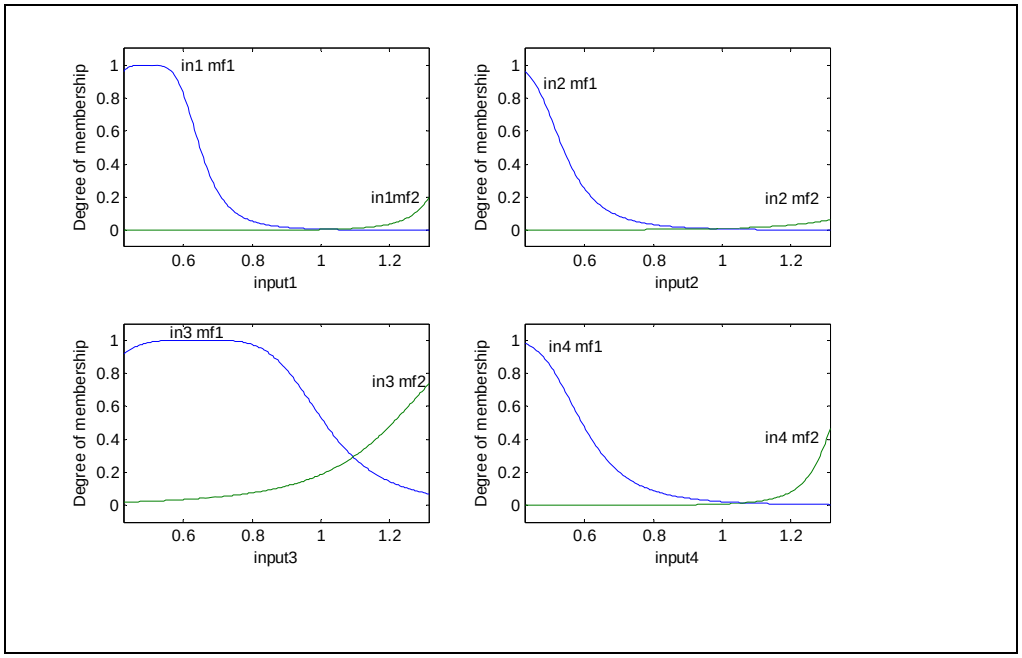
n = the number of membership function where $n \geq 2$,
r = the range of FANFIS input,
m = the minimum value of FANFIS input,
$i = 0$ to $n-1$,

then the parameters for the $(i+1)^{\text{th}}$ generalized bell membership function are

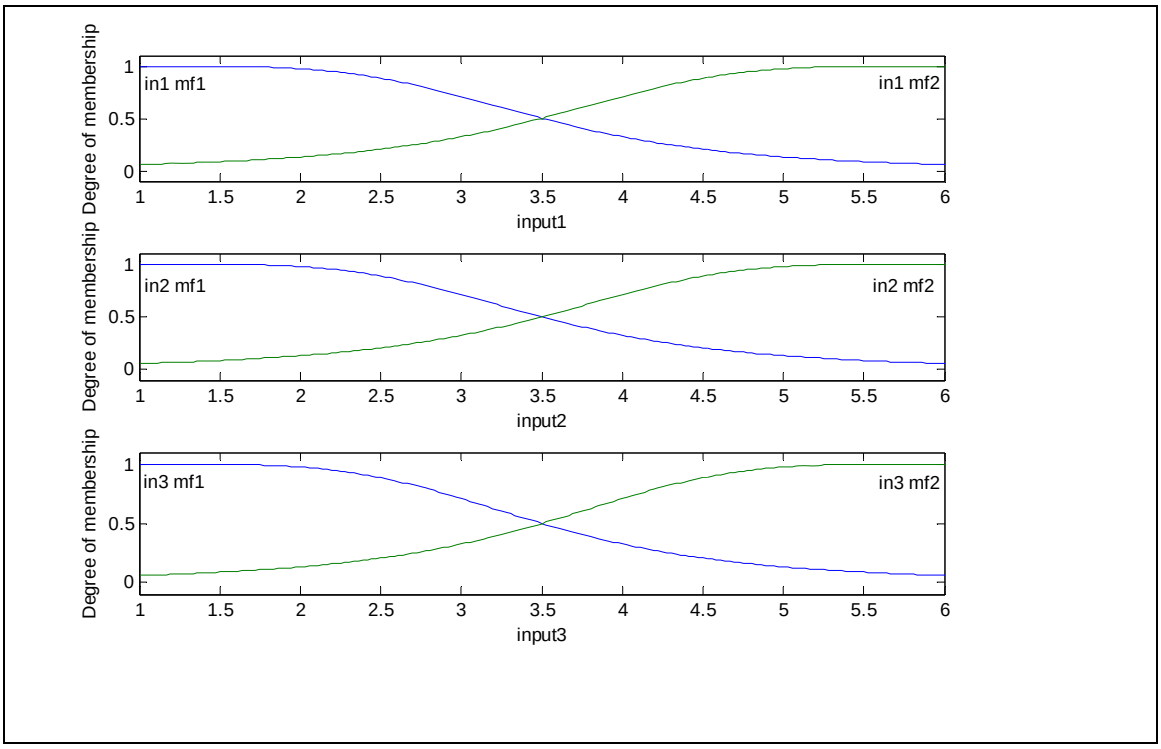
$$a=\left(\frac{0.5r}{n-1}\right), b=2 \text{ and } c=m+i\left(\frac{r}{n-1}\right).$$



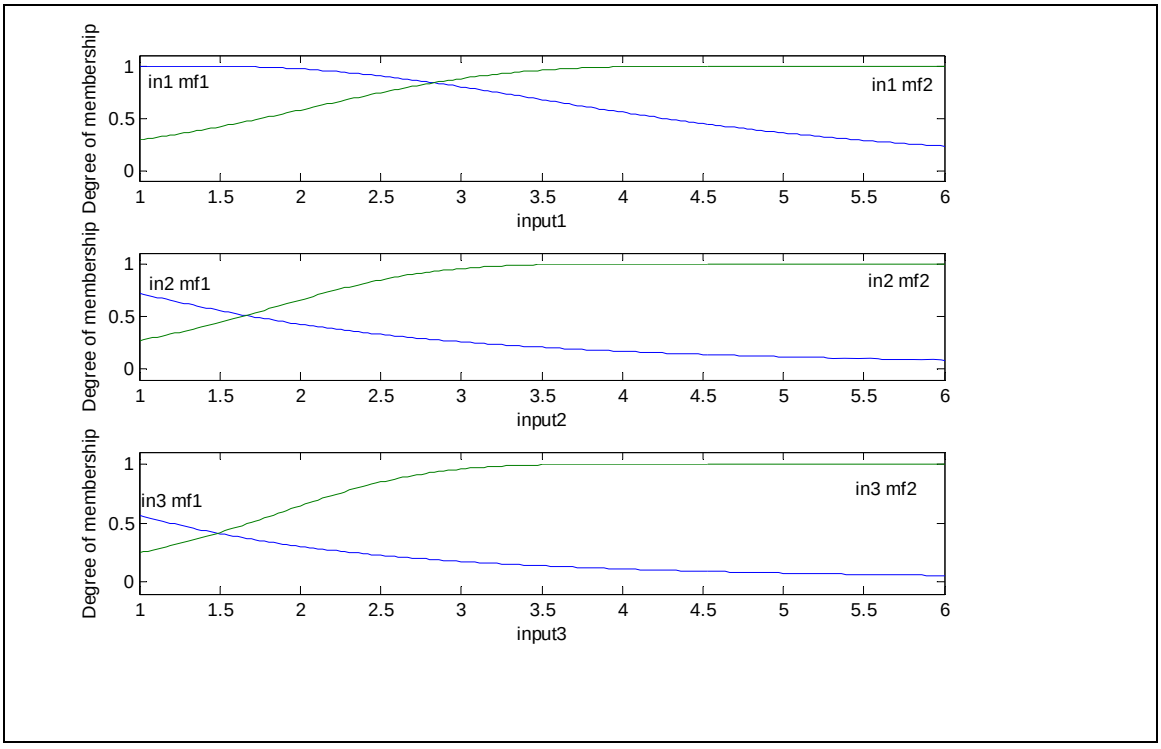
Initial membership functions in Section 7.3.1



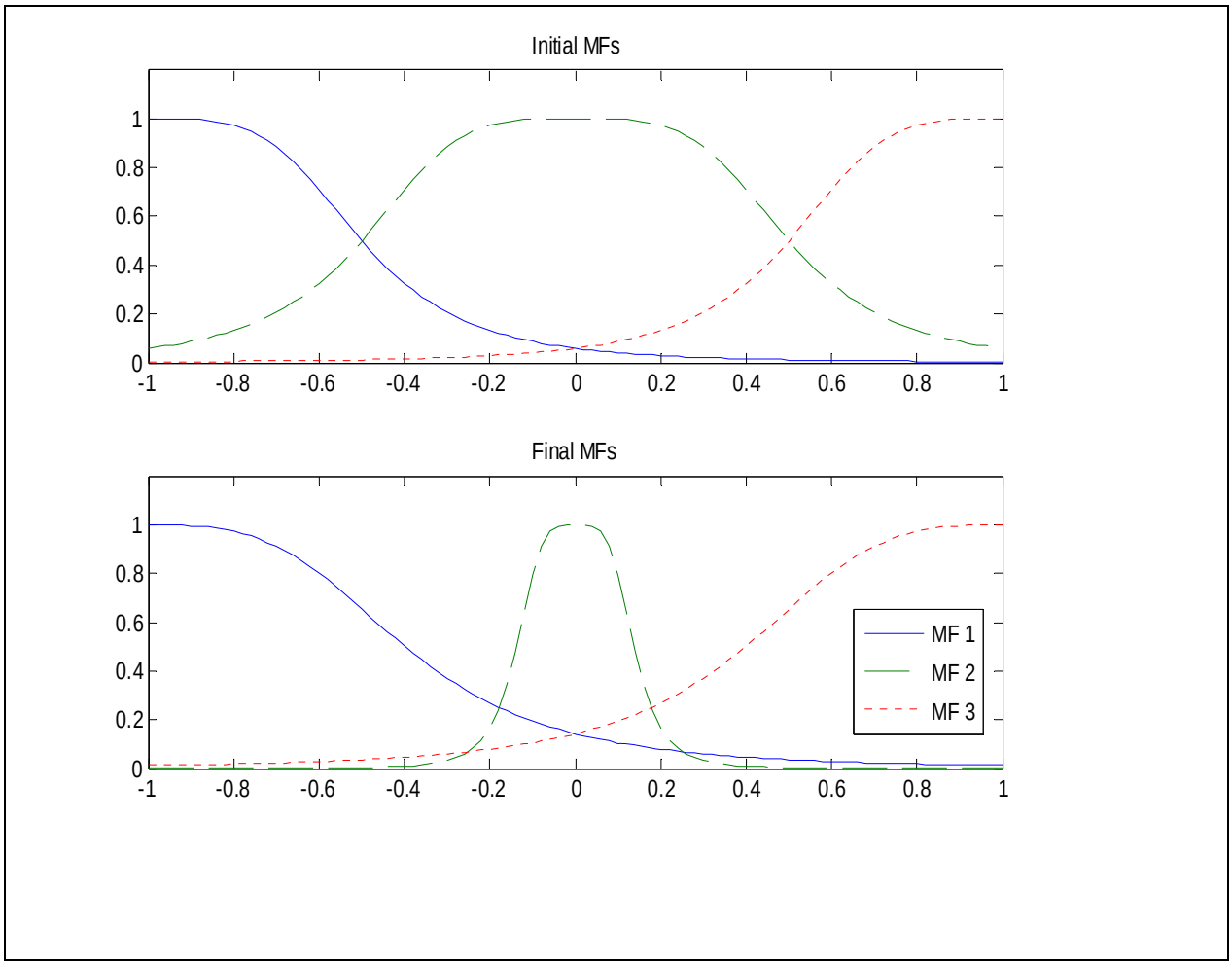
Final membership functions in Section 7.3.1



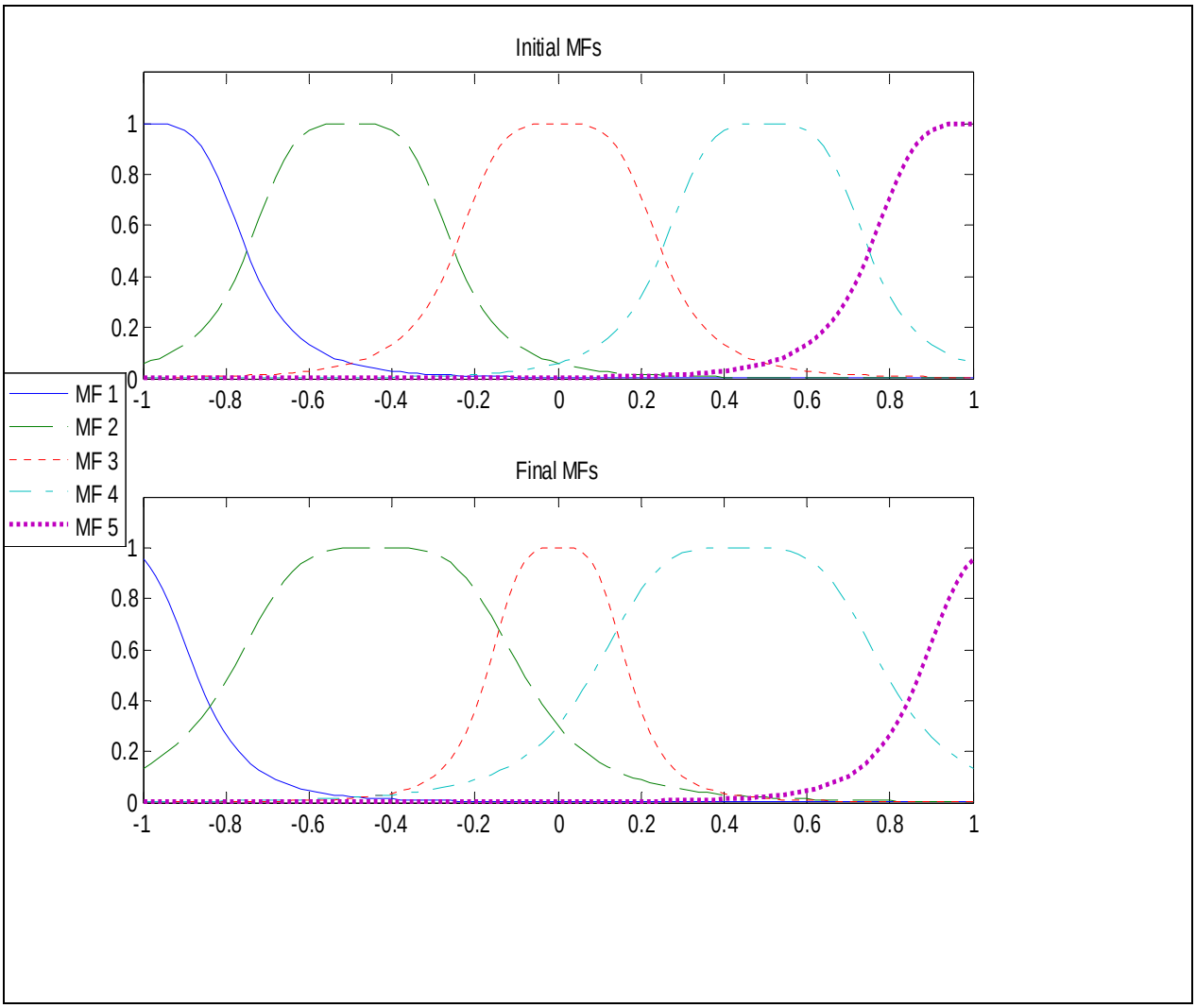
Initial membership functions in Section 7.3.2



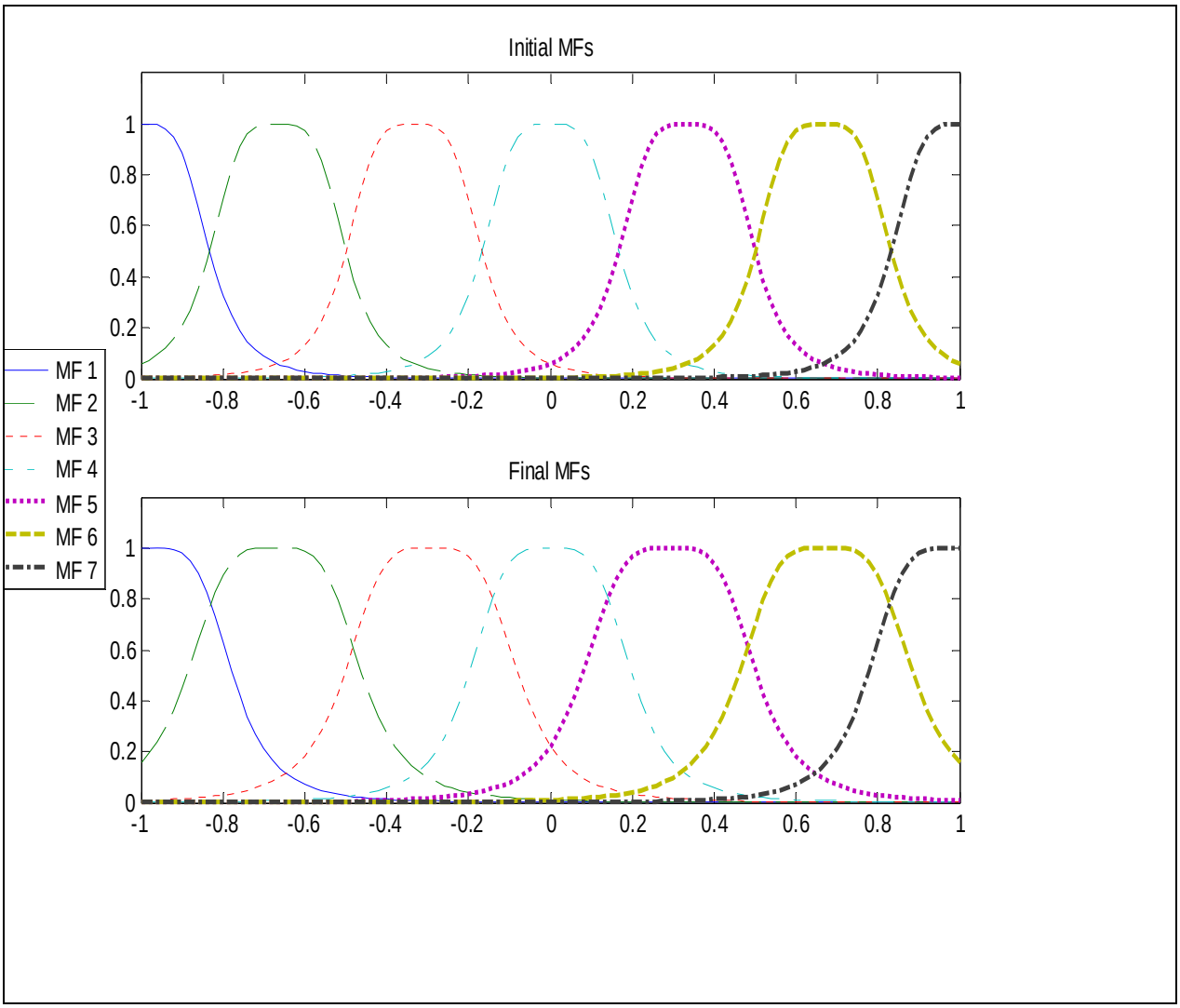
Final membership functions in Section 7.3.2



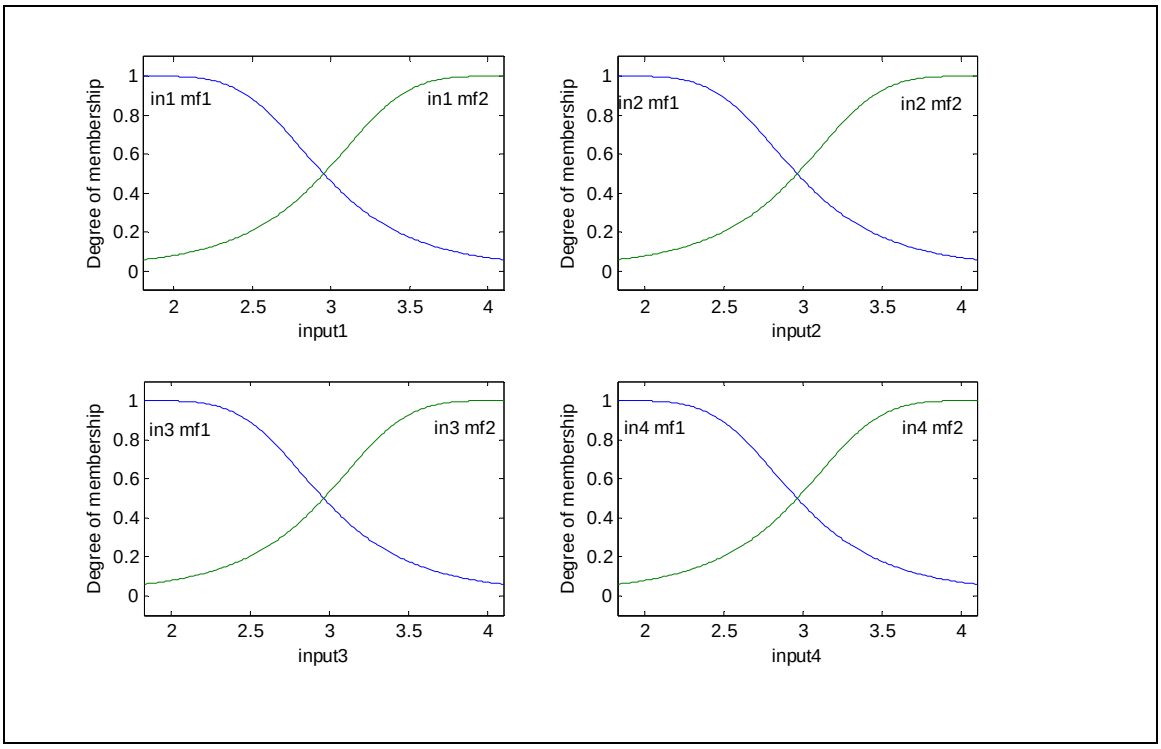
Initial and final membership functions in Section 7.3.3 [3 Membership Functions case]



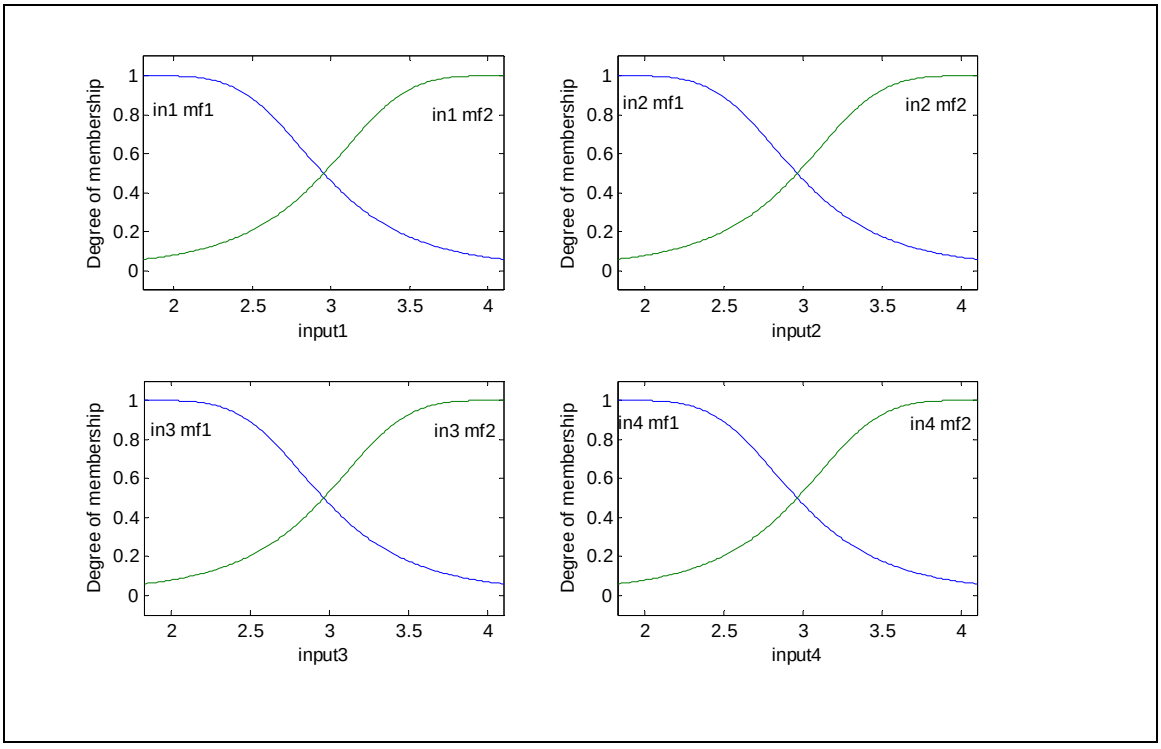
Initial and final membership functions in Section 7.3.3 [5 Membership Functions case]



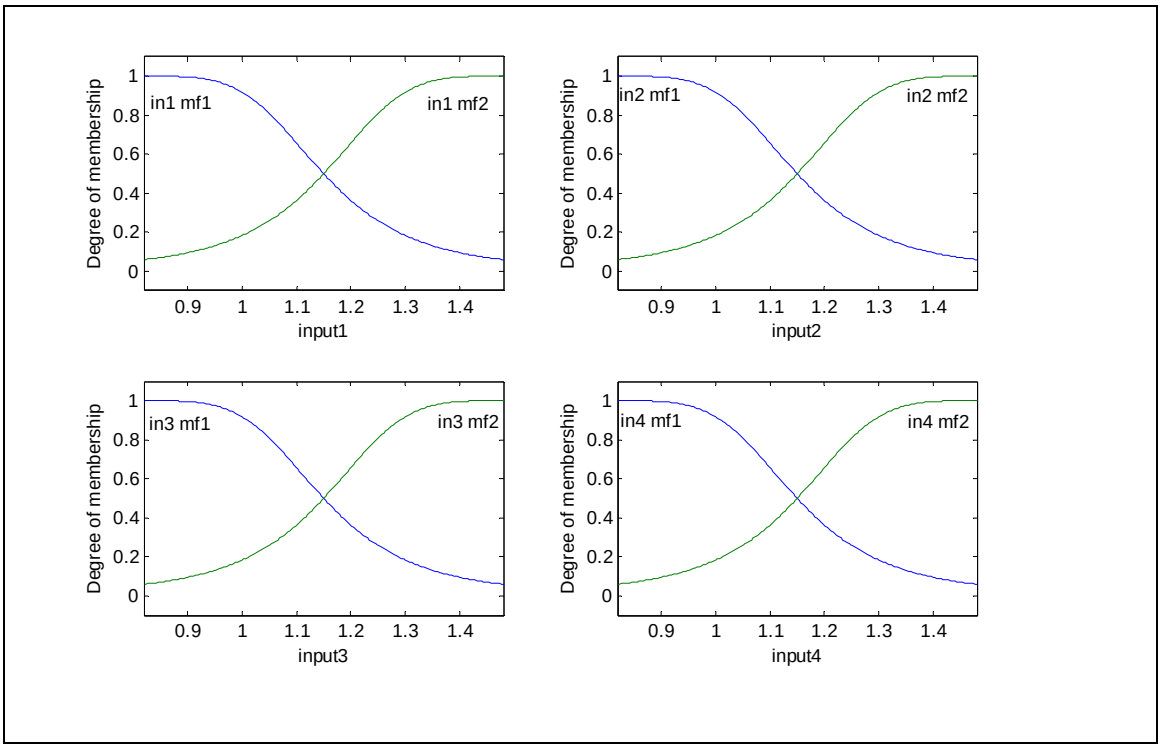
Initial and final membership functions in Section 7.3.3 [7 Membership Functions case]



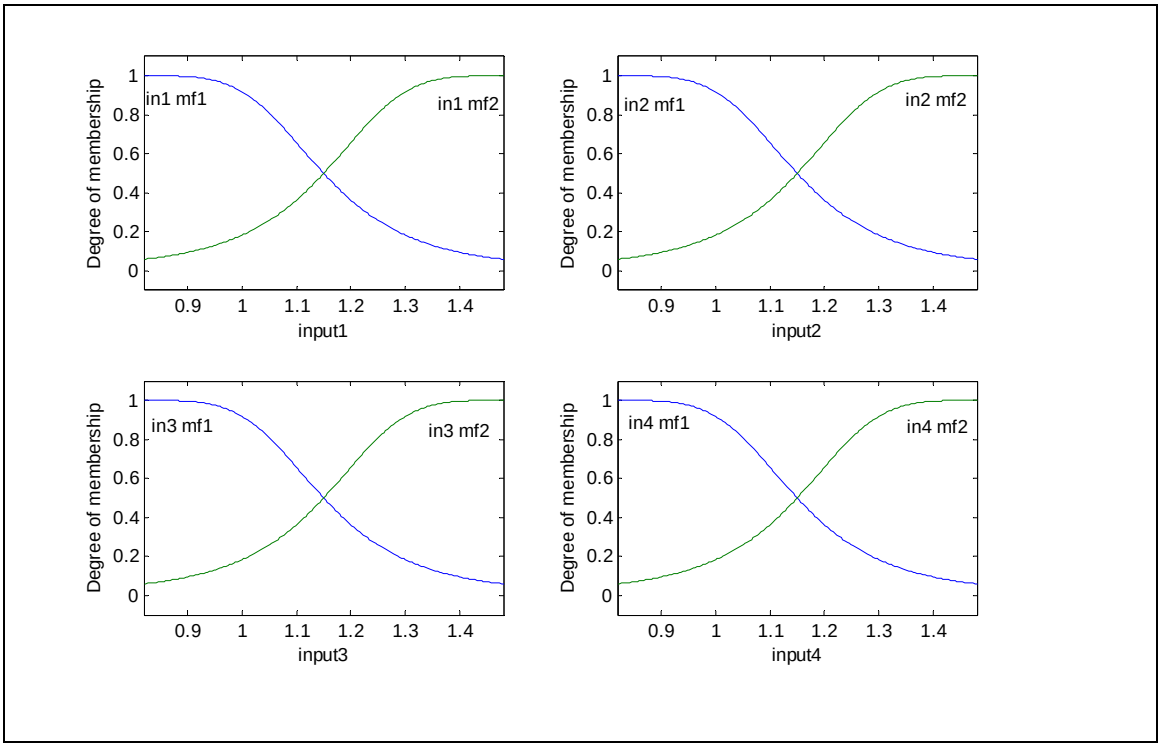
Initial membership functions in Section 7.3.4 [Air New Zealand "A"]



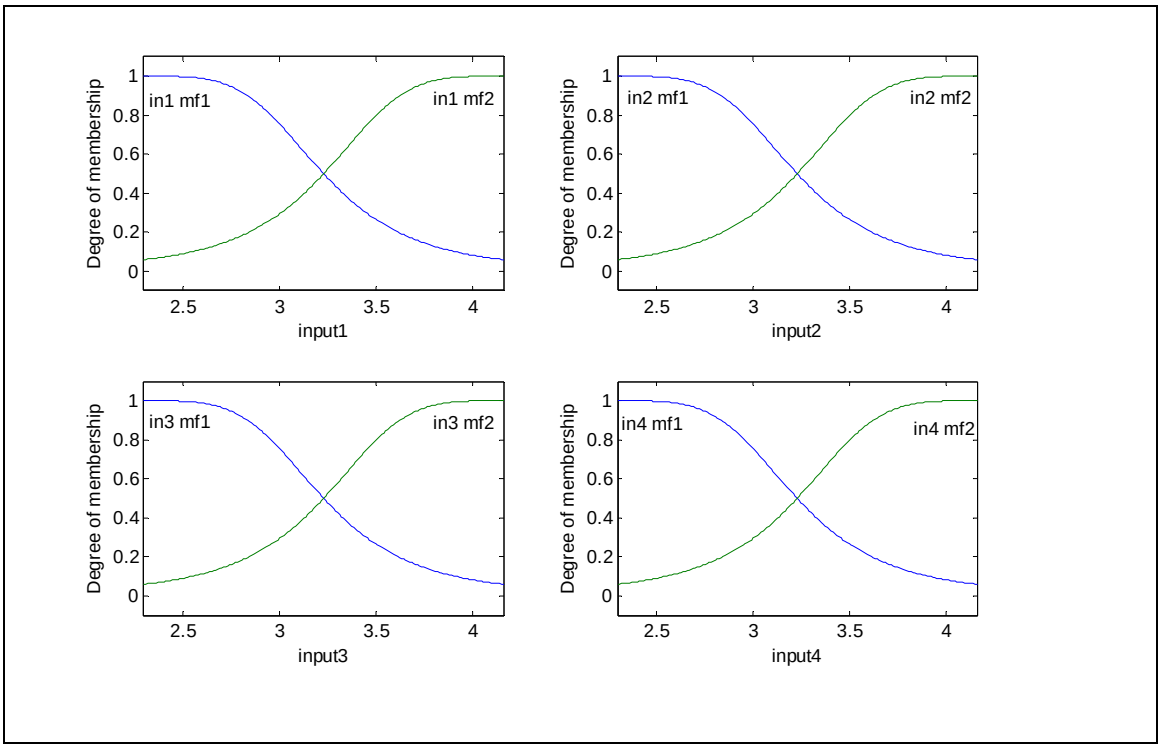
Final membership functions in Section 7.3.4 [Air New Zealand "A"]



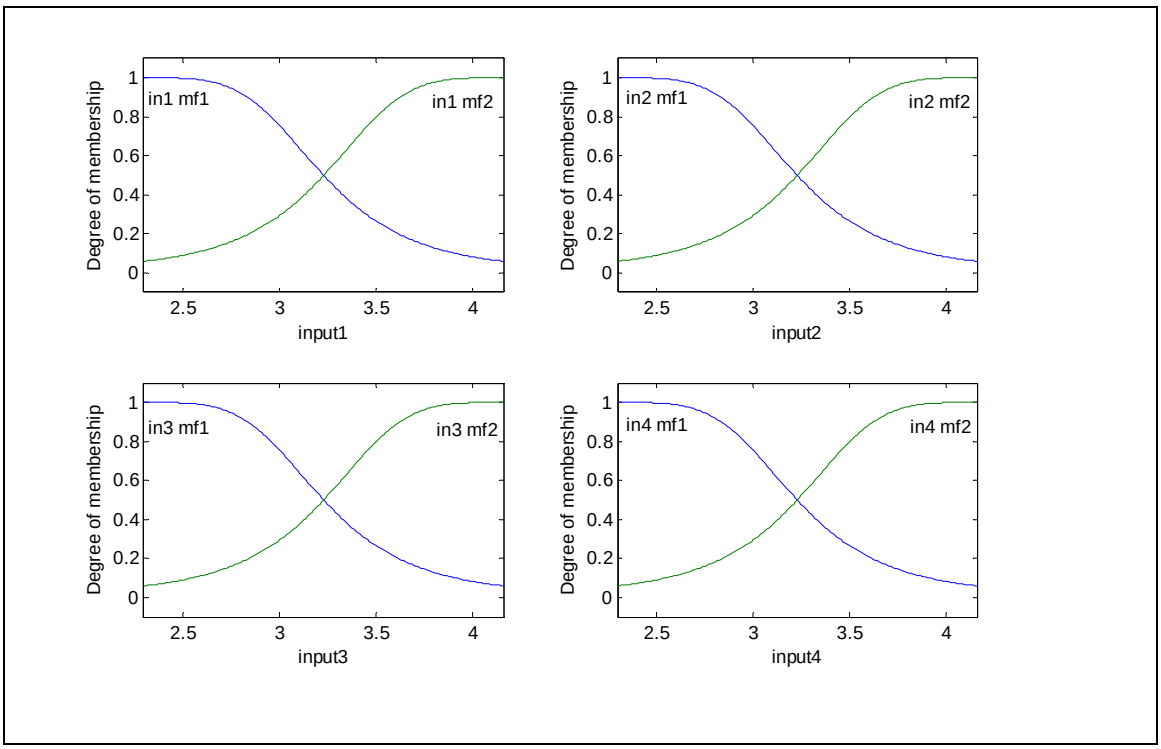
Initial membership functions in Section 7.3.4 [Brierley]



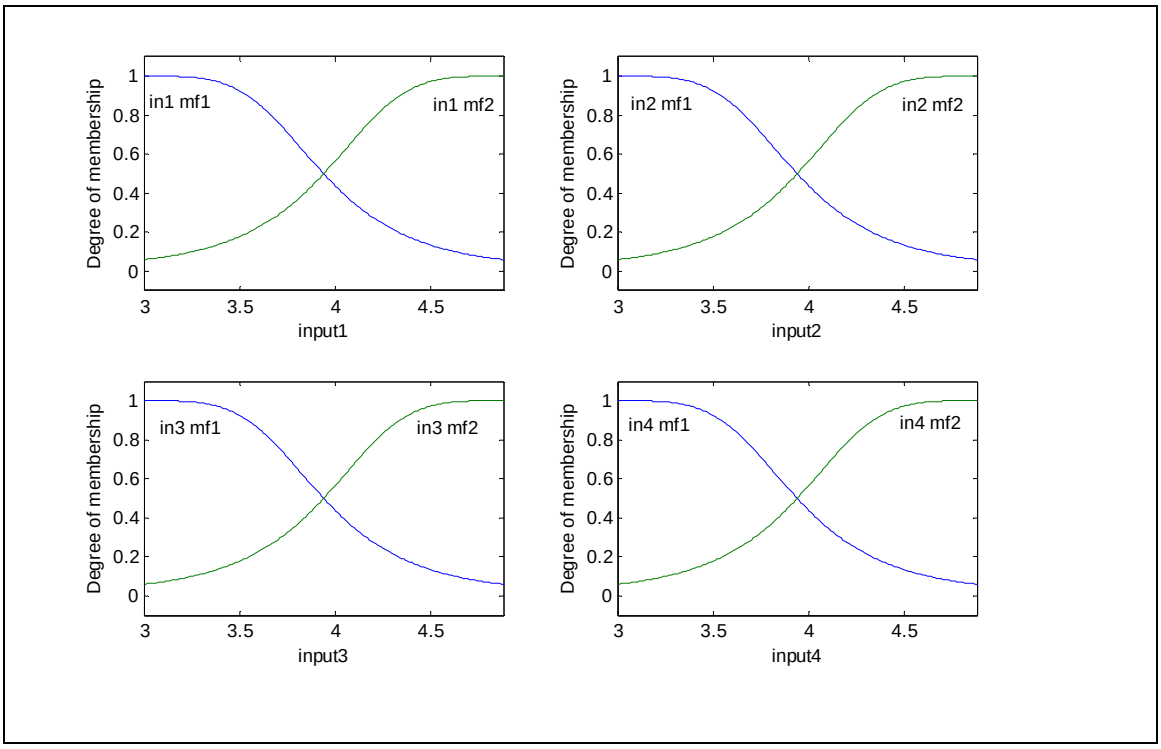
Final membership functions in Section 7.3.4 [Brierley]



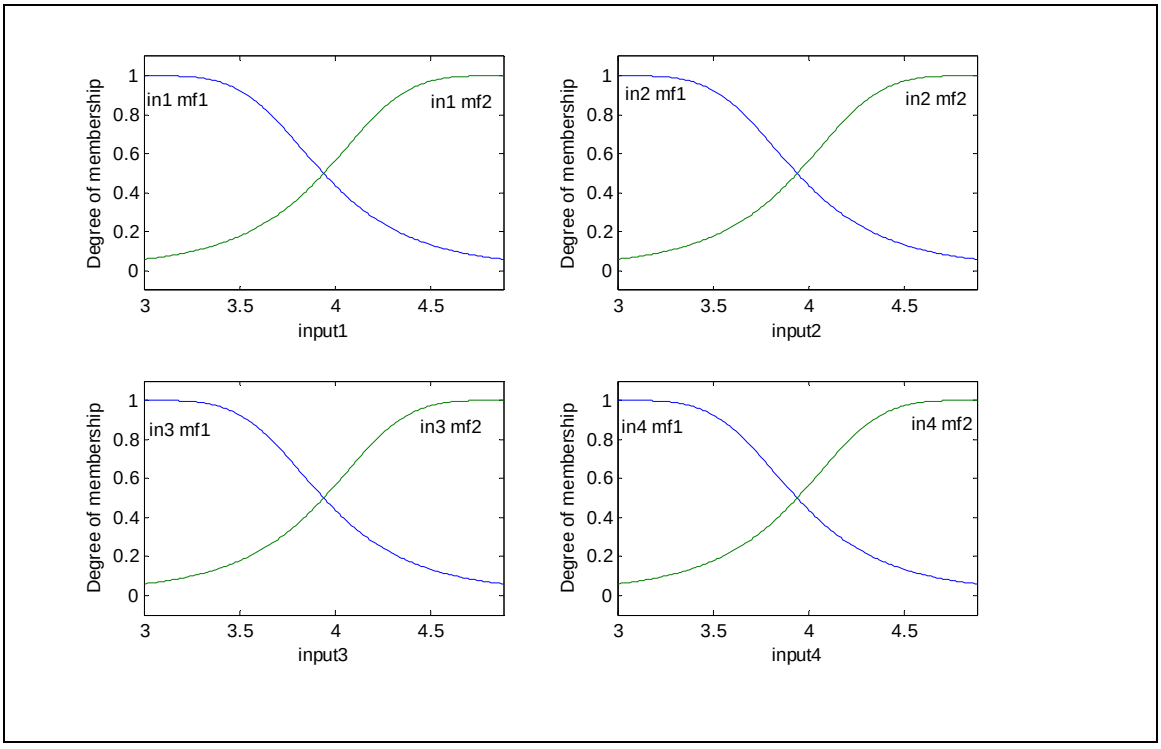
Initial membership functions in Section 7.3.4 [Carter Holt Harvey]



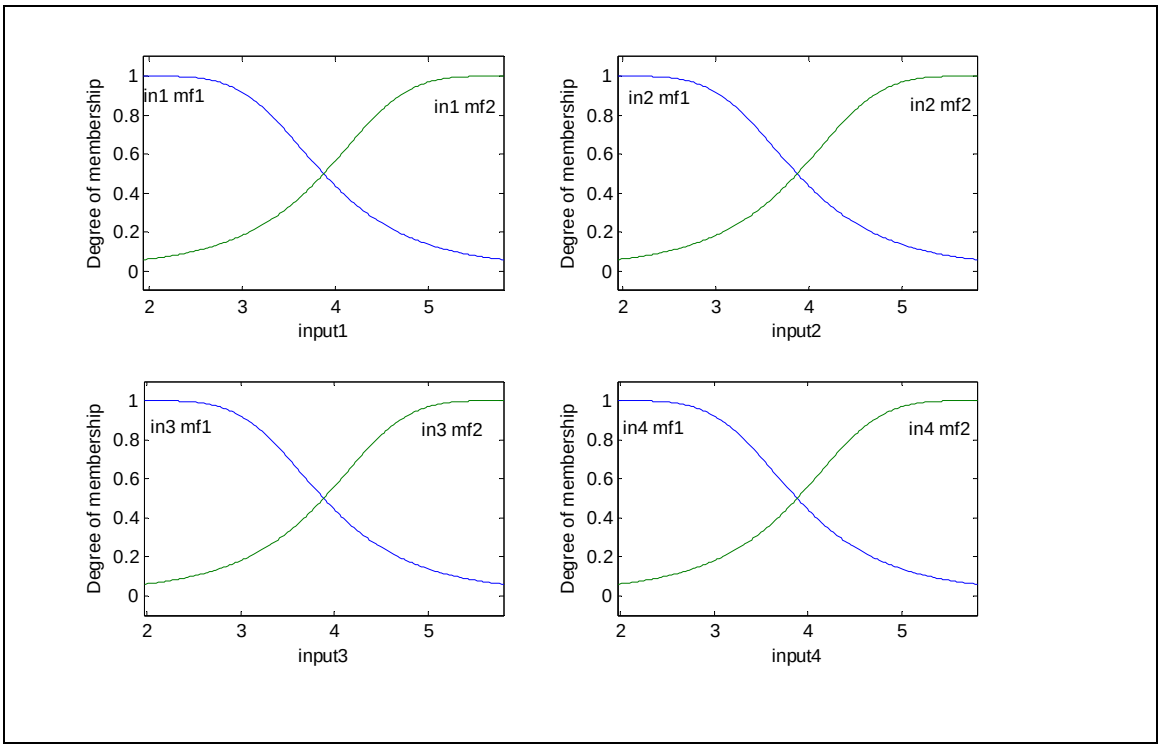
Final membership functions in Section 7.3.4 [Carter Holt Harvey]



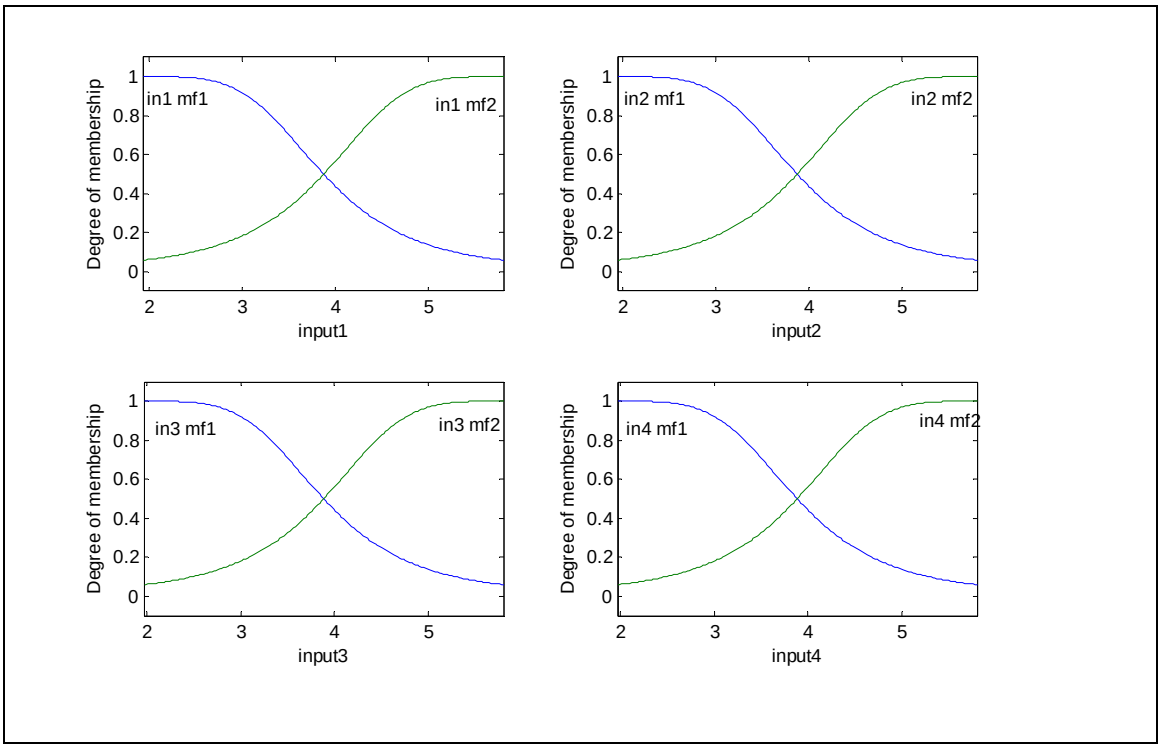
Initial membership functions in Section 7.3.4 [Lion Nathan]



Final membership functions in Section 7.3.4 [Lion Nathan]



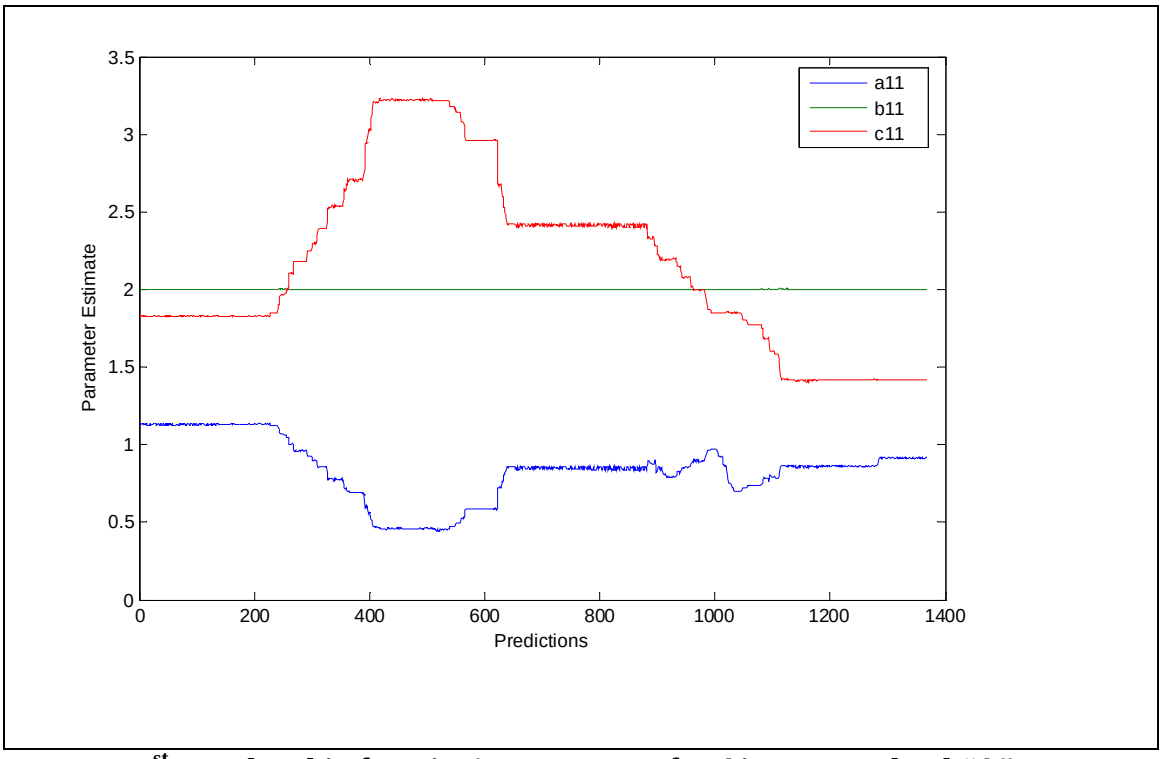
Initial membership functions in Section 7.3.4 [Telecom]



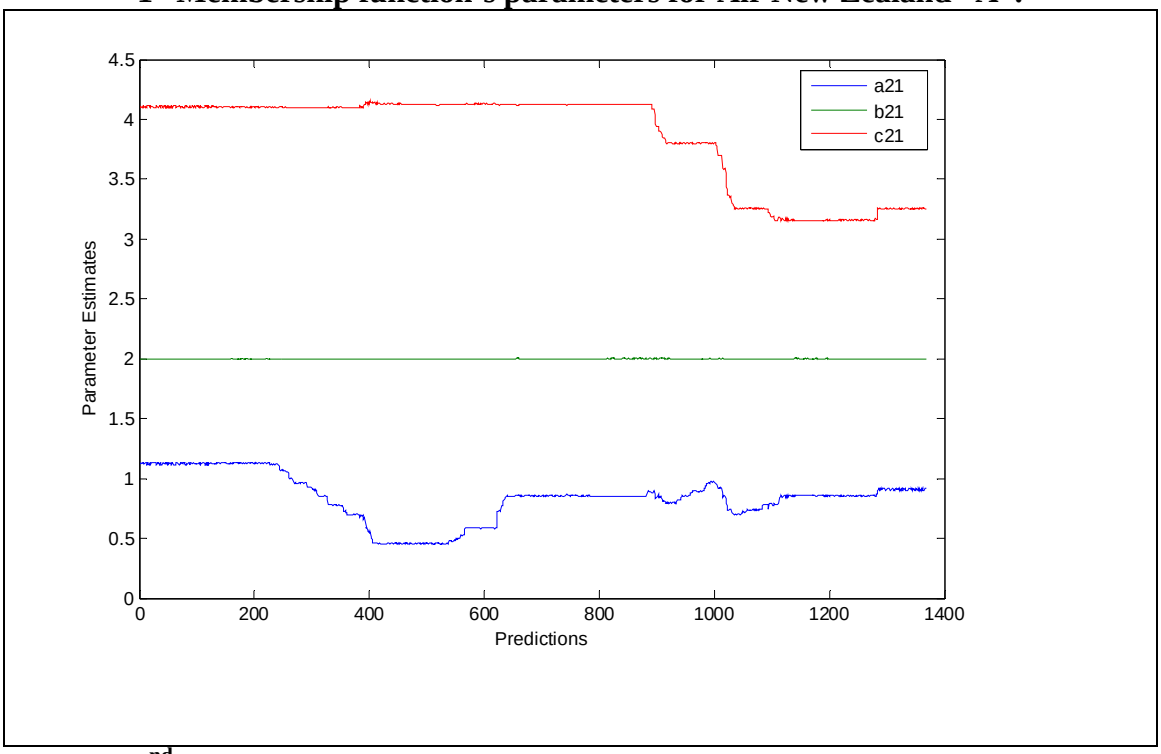
Final membership functions in Section 7.3.4 [Telecom]

Appendix II

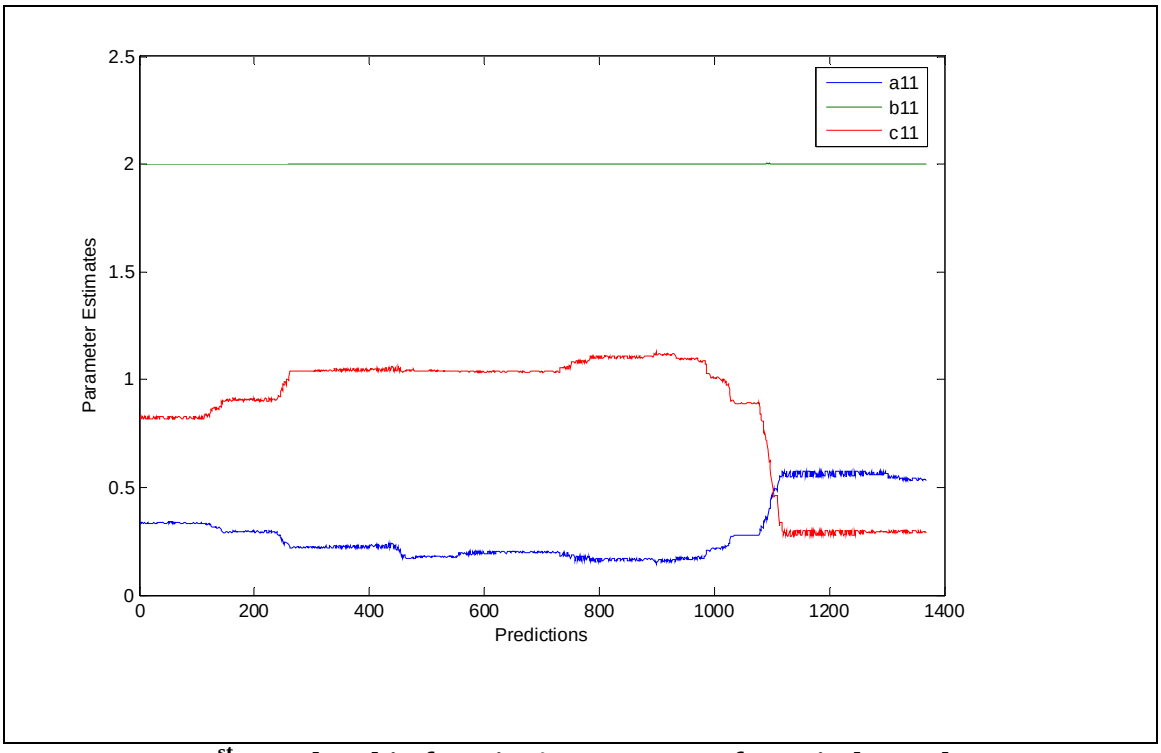
Sensitivity of nonlinear parameter estimates in Section 7.3.4



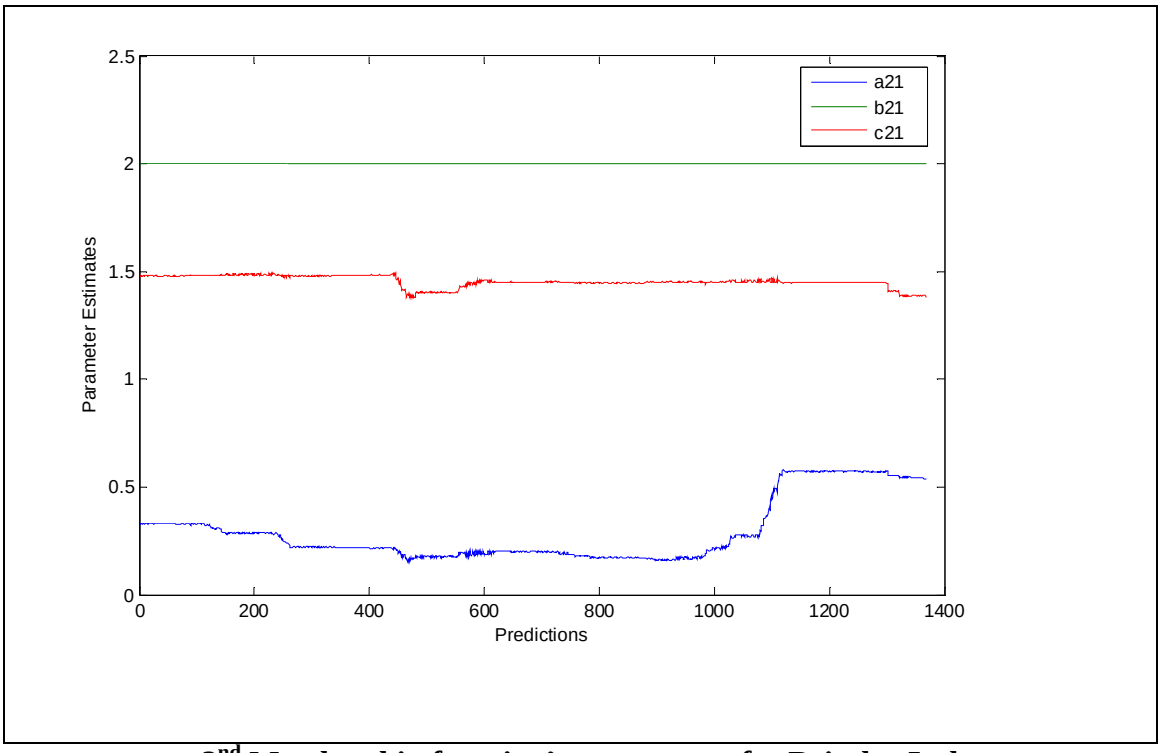
1st Membership function's parameters for Air New Zealand "A".



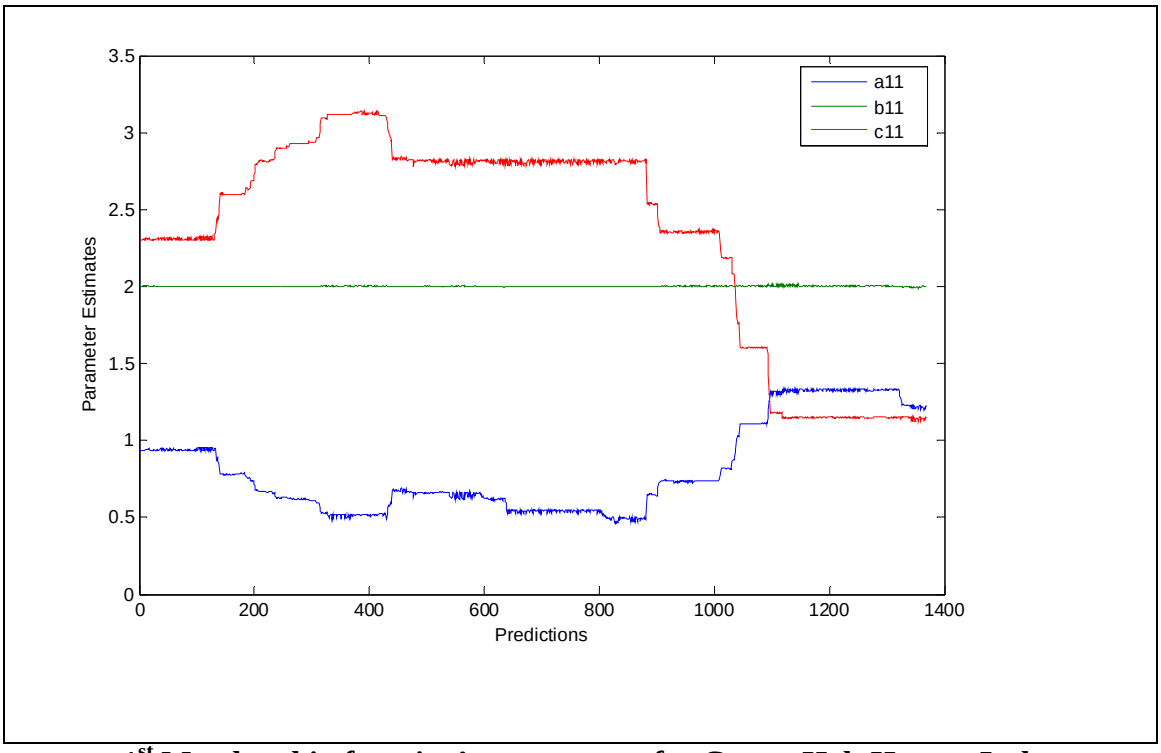
2nd Membership function's parameters for Air New Zealand "A".



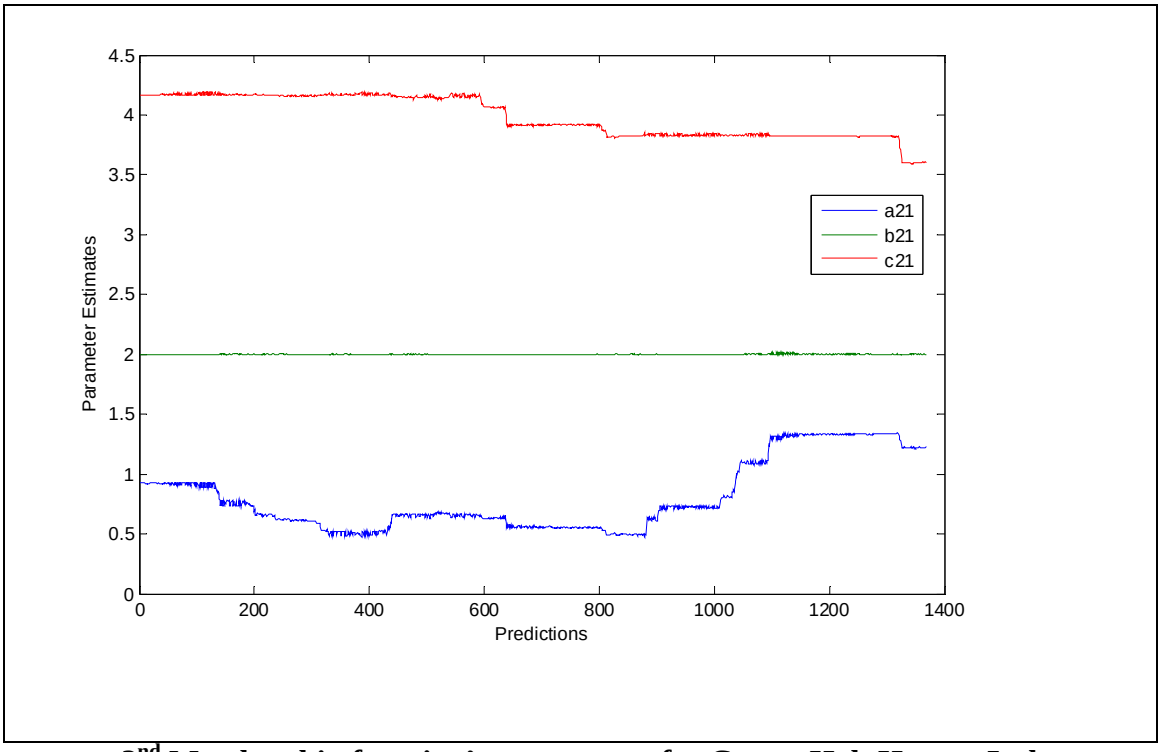
1st Membership function's parameters for Brierley Ltd.



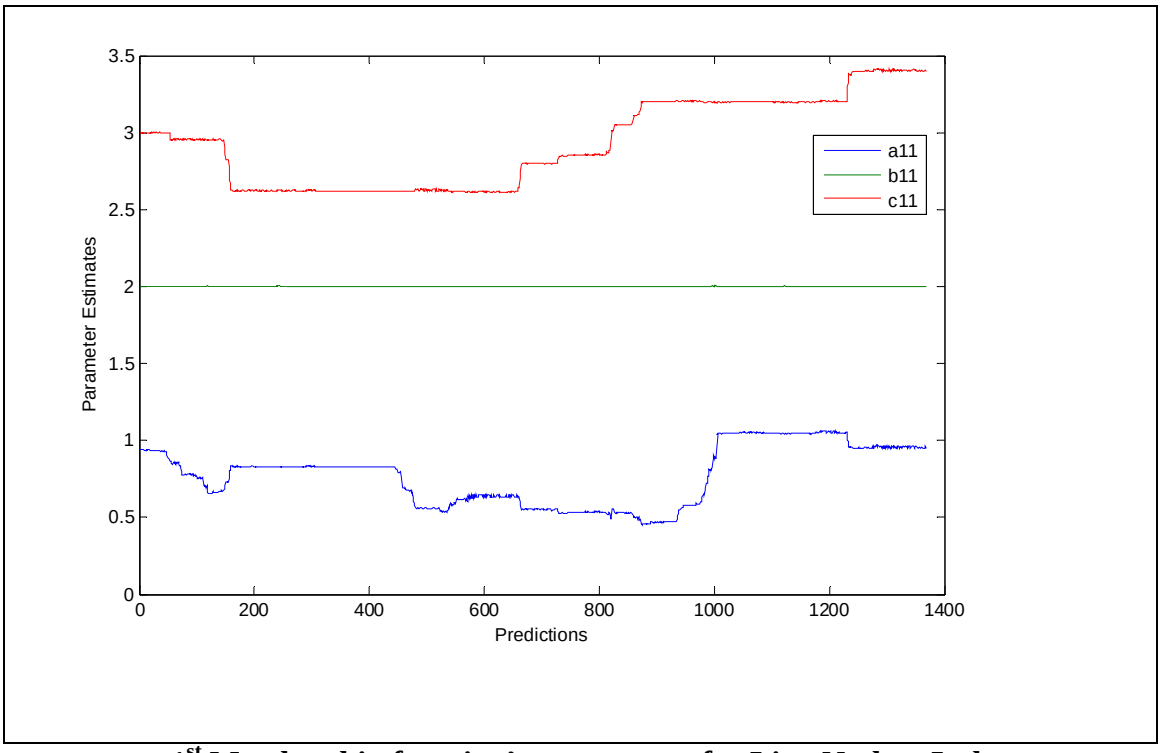
2nd Membership function's parameters for Brierley Ltd.



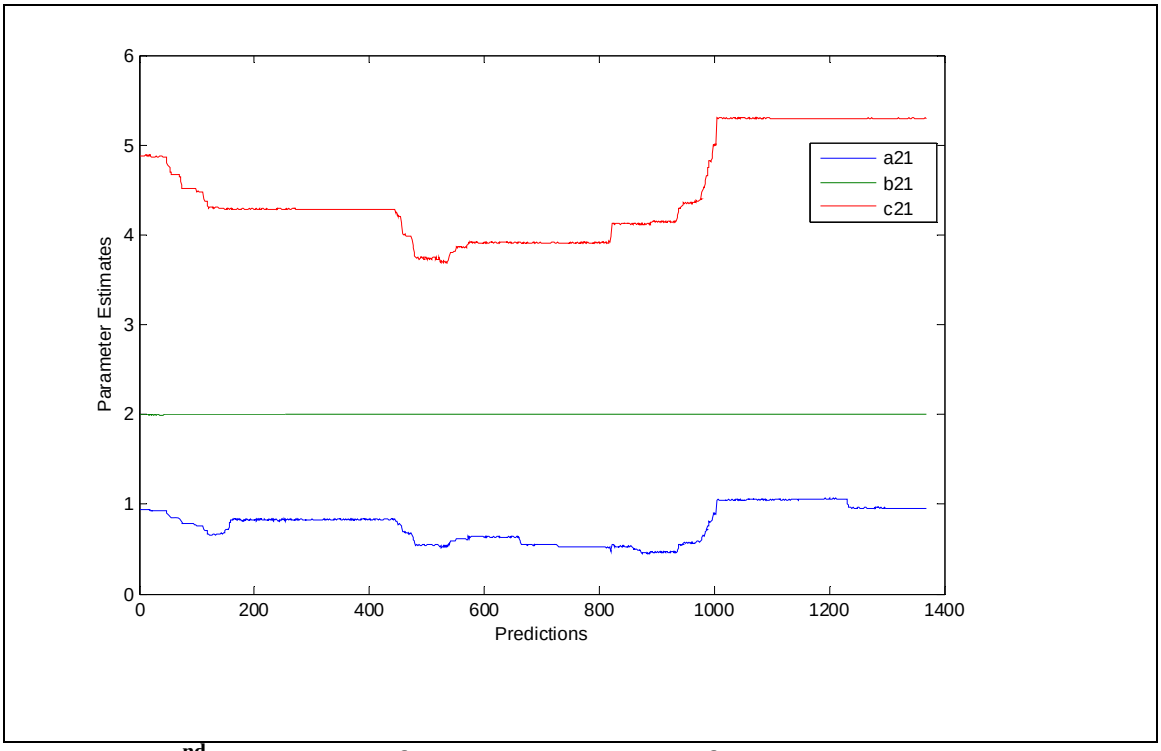
1st Membership function's parameters for Carter Holt Harvey Ltd.



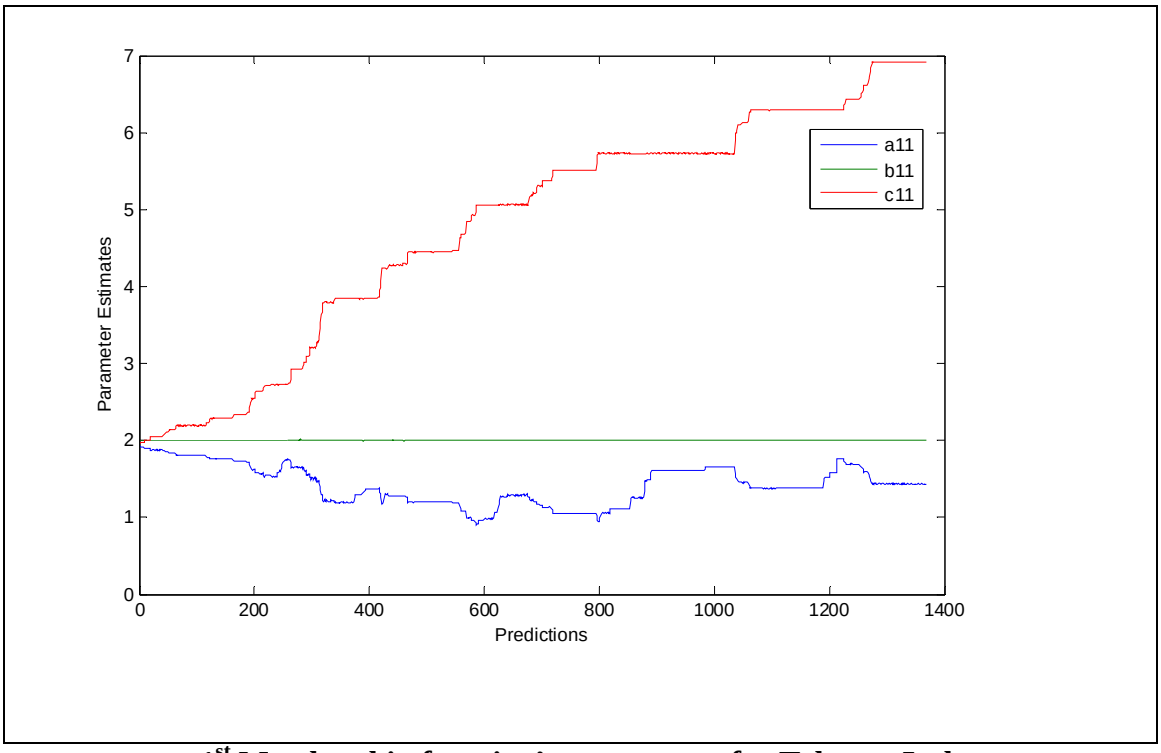
2nd Membership function's parameters for Carter Holt Harvey Ltd.



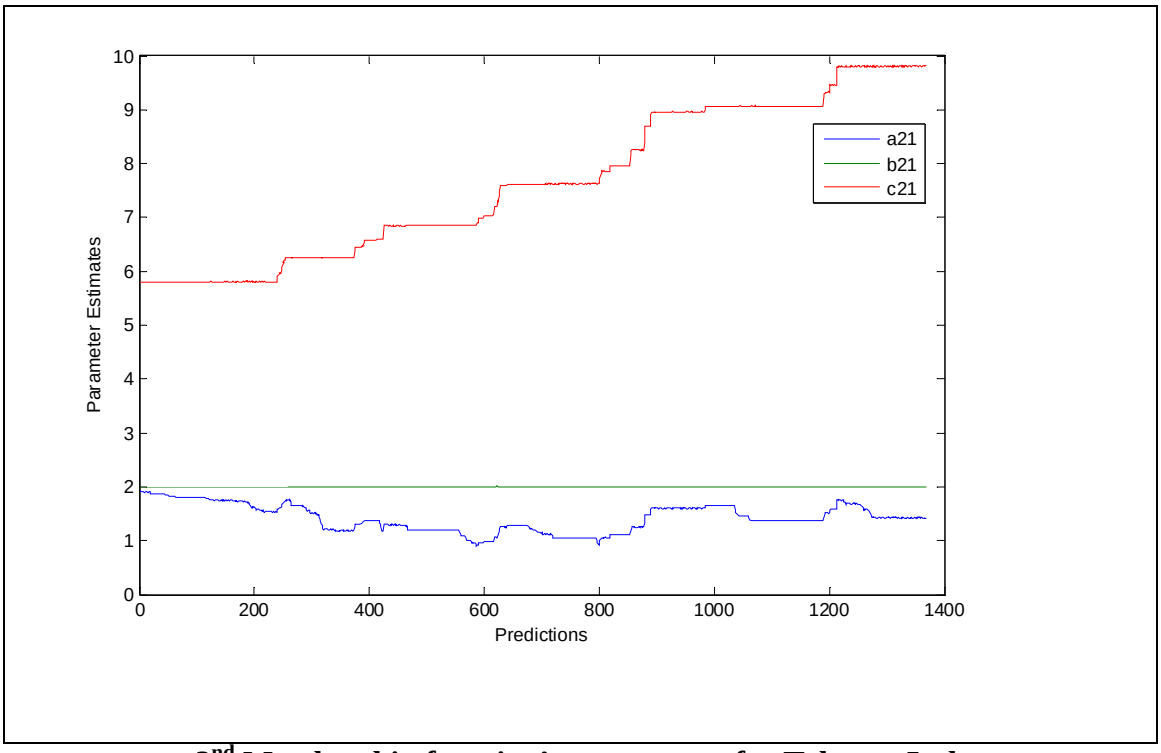
1st Membership function's parameters for Lion Nathan Ltd.



2nd Membership function's parameters for Lion Nathan Ltd.



1st Membership function's parameters for Telecom Ltd.



2nd Membership function's parameters for Telecom Ltd.

References

1. J. S. Roger Jang. ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. On Systems, Man and Cybernetics,* 23(03):665-668, May 1993.

2. T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. On Systems, Man and Cybernetics*, 15:116-132, 1985.

3. D. E. Rumelhart, G.E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1,chapter 8, pages 318-362. The MIT Press*, 1986.

4. P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Harvard University*, 1974.

5. M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science,* 197:287-289, July 1977.

6. L. A. Zadeh. Fuzzy sets, *Information and Control* 8, 338-353, 1965.

7. J. Lukasiewicz. O logice trójwartociowej, *Ruch Filozoficzny* 5, 170f, 1920.

8. L. A. Zadeh. A rationale for Fuzzy Control, *Journal of Dynamic Systems, Measurement, Control* 94 (6), 3-4, 1972.

9. L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, Parts 1, 2 and 3. *Information Sciences,* 8:199-249, 8:301-357, 9:43-1975.

10. L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on System, Man and Cybernetics*,3(1):28-44, January 1973.

11. E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-achine Studies,* 7(1):1-13, 1975.

12. S. Fukami, M. Mizumoto, and K. Tanaka. Some considerations on fuzzy conditional inference. *Fuzzy Sets and Systems,* 4:243-273, 1980.

13. Y. Tsukamoto. An approach to fuzzy reasoning method. In Madan M. Gupta, Rammoha K. Ragade, and Ronald R. Yager, editors, *Advances in fuzzy set theory and applications*, pages 137-149. North-Holland, Amsterdam, 1979.

14. J. J. Buckley. Universal fuzzy controllers. Automatica, 28:1245-1248, 1992.

15. A. Kandel, editor. *Fuzzy expert systems.* CRC Press, Inc., Boca Raton, FL, 1992.

16. B. Kosko. *Neural networks and fuzzy system: a dynamical systems approach.* Prentice Hall, Upper Saddle River, NJ, 1991.

17. N. Pfluger, J. Yen, and R. Langari. A defuzzification strategy for a fuzzy logic controller employing prohibitive information in command formulation. In *proceeding of IEEE International Conference on Fuzzy Systems, pages 717-723,* San Diego, March 1992.

18. T. A. Runker and M. Glesner. Defuzzification and ranking in the context of membership value semantic, rule modality, and measurement theory. In *European Congress on Fuzzy and Intelligent Technologies,* Aachen, September 1994.

19. R. R. Yager and D. P. Filev. SLIDE: A simple adaptive defuzzification method. *IEEE Transactions on Fuzzy Systems,* 1(1):69-78, February 1993.

20. J. S. Roger Jang and C. T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE transactions on Neural Networks,* 4(1):156-159, January 1993.

21. A. E. Bryson and Y. C. Ho. *Appiled optimal control.* Blaisdell, New York, 1969.

22. D. B. Parker. Learning logic. Invention report S81-64, File 1, Office of technology Licensing, Standford University, October 1982.

23. J. S. Roger Jang. Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91),* pages 762-767, July 1991.

24. N. J. Nilsson. *Learning machines: foundation of trainable pattern classifying systems.* McGraw-Hill, New York, 1965.

25. F. Rosenblatt. *Principle of Neurodynamics: Perceptrons and the theory of brain mechanisms.* Spartan, New York, 1962.

26. C. T. Lin and C. S. G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers,* 40(12):1320-1336, December 1991.

27. L. X. Wang and J. M. Mendel. Back-propagation fuzzy system as nonlinear dynamic system identifiers. In *Proceedings of the IEEE International Conference on Fuzzy systems*, San Diego, March 1992.

28. C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate distribution. In *Proc. Third Berkeley Symp. Math. Statist. Prob.*: **1**, 197-206, 1956.

29. B. Efron and C. Morris. Stein's paradox in statistics. *Scientific American,* **238** (5): 119-127, 1977.

30. A. S. Lapedes and R. Farber. Non-linear signal processing using neural networks: prediction and system modelling Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, 1987.

31. J. Moody. Fast learning in multi-resolution hierarchies. In D. S. Touretzky, editor, *Advance in neural information processing systems I*, chapter 1, pages 29-39. Morgan Kaufmann, San Mateo, CA, 1989.

32. J. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation,* 1:281-294, 1989.

33. R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, and P. S. Lewis. Function approximation and time series prediction with neural networks. In *Proceedings of IEEE international Joint Conference on Neural Networks,* pages 649-665 (Volume I), 1990.

34. R. S. Crowder III. Predicting the Mackey-Glass time series with cascade-correlation learning. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 117-123, Carnegie Mellon University, 1990.

35. T. D. Sanger. A tree-structured adaptive network for function approximate in high dimensional spaces. *IEEE Transactions on Neural Networks*, 2(2): 285-293, March 1991.

36. H. Takagi and I. Hayashi. NN-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 5(3):191-212, 1991.

37. M. Sugeno and G. T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28:15-33, 1988.

38. T. Kondo. Revised GMDH algorithm estimating degree of the complete polynomial. *Trans. of the Society of Instrument and Control Engineers*, 22(9):928-934, 1986.

39. K. S. Narendra and K. Parthsarathy. Identification and control of dynamical systems using neural networks. *IEEE Transaction on Neural Networks*, 1(1):4-27, 1990.

40. I. Weeraprajak. A comparative study of Time-series forecasting applied to stock market price. *Master thesis, University of Canterbury*, 2001.

41. G. E. P. Box and G. M. Jenkins. Some Statistical Aspects of Adaptive Optimization and Control, *Journal of the Royal Statistical Society* B 24 (2), 279-331, 1962.

42. P. J. Harrison and C. F. Stevens. Bayesian forecasting, *Journal Royal Statistics Society* B 38, 205-228, 1976.

43. A. Hobbs and N. G. Bourbakis. A neurofuzzy arbitrage simulator for stock investing, *Proc. IEEE/IAFE Conf. Computational Intelligence Financial Engineering,* New York, April 9-11, 1995.

44. E. F. Fama. Efficient capital markets: A review of theory and empirical work, *Journal of Finance* 25, 383-417, 1970.

45. C. W. J. Granger. Empirical studies of capital markets: A survey, in: eds., G. Szego and K. Shell, *Mathematical Methods in Investment and Finance.* North-Holland: Amsterdam, 1972.

46. M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 586-591, San Francisco, 1993.

47. M. C. Jensen. Some anomalous evidence regarding market efficiency, *Journal of Financial Economics* **6**, 95-101, 1978.

48. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and Regression Trees.* Wadsworth International, Belmont, Ca, 1984.

49. B. Kosko. Fuzzy systems as universal approximators. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 1153–1162, San Diego, 1992.

50. L. X. Wang. Fuzzy systems are universal approximators. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 1163–1169, San Diego, 1992.